Conjunctive Query Answering over Unrestricted OWL 2 Ontologies

Federico Igne

Linacre College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2022

A Davide

Acknowledgements

I would like to express my most sincere gratitude to my supervisors, Professor Ian Horrocks and Dr Stefano Germano, for their guidance and support in the past years. My thanks go, in particular, to Prof. Horrocks for being an invaluable source of advice, never failing to share his experience in conducting research and for helping me prioritize and stay focused on the final goal. Special thanks go to Stefano, for his dedication in helping me navigate this DPhil, for our regular meetings and precious help.

My gratitude goes also to the SIRIUS Research Centre, for funding my research and for all the opportunities for personal and professional growth that they offered me in these past years.

My experience, here in Oxford, would not have been the same without all the lovely people I had the opportunity to meet. A big “thank you” goes to all members of the KRR group, SIRIUS research centre, the OxRAM society, and past and present Oatlanders. Honourable mentions go to Alessandro and Babis, for our coffee breaks by the sofa; to Przemek, for being the perfect gym buddy who inevitably turned into a whisky tasting buddy; to Alessio, for making sure I wouldn’t forget my origins; to Stefano, for our “off-topic” chats whose backlog is longer than this work; to David, for being a constant reference point during my DPhil, and for being the companion I’d choose if I ever had to delve in a dungeon; to Max, co-host of the Monday Rants, a podcast about the joy of research, long-distance relationships and everything in between. . . you have been a true friend.

Infine, la mia più profonda gratitudine va ai miei genitori per essere, ancora oggi, punto di riferimento sicuro e sostegno nei momenti di difficoltà, a Riccardo, da sempre compagno di giochi, e a tutta la mia famiglia. Un grazie speciale va a Giulia, che in tutti questi anni mi è rimasta vicina, anche nella distanza, e ha sempre assecondato le mie passioni più disparate, sin da quando portavo il libro di λ-calcolo in vacanza. Ora torno a casa.

Abstract

Conjunctive query (CQ) answering is one of the primary reasoning tasks over knowledge bases (KBs). However, when considering expressive description logics (DLs), 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 black-box services to provide a CQ answering service for OWL (Web Ontology Language). 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. These techniques led to the development of two new systems: (i) RSAComb, an efficient implementation of a new tractable answering service for the RSA (role safety acyclic) ontology language; (ii) ACQuA, a reference implementation of the proposed hybrid architecture combining RSAComb, PAGOdA (Zhou, Cuenca Grau, Nenov, et al. 2015), and HermiT (Glimm, Horrocks, Motik, et al. 2014) 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 showed how ACQuA can reliably match PAGOdA's performance and further limit its performance issues, especially when the latter extensively relies on the underlying fully-fledged reasoner.

Contents

List of Figures

List of Tables

1
Introduction

Efficient data management and data access have become a primary problem in the design and development of applications, especially due to the large amount of data we produce every day.

Description logics (DLs) [3, 4] are a logic-based formalism for knowledge representation (KR) and reasoning, dating back to the late 1980s. They can be used to effectively model a certain domain of interest in a structured and well-defined way; this is done by defining the foundational notions of a domain in terms of *individuals* (entities), *concepts* (classes of entities) and *roles* (relation between entities). DLs are formally defined as decidable fragments of First-Order (FO) logic, and this very connection to a logical formalism is what makes them differ from previous attempts at including knowledge into intelligent systems (e.g., *semantic networks* [90], *frames* [71]).

Domain knowledge is usually divided into two major components, a *terminological* part, called TBox (or *ontology*), and an *assertional* part, called ABox. The ABox represents explicit knowledge about the domain in terms of known facts, while the TBox is used to represent the structure of the domain and the rules governing it. The combination of a TBox and an ABox is called a knowledge base (KB).

Common reasoning tasks performed over KBs include checking for satisfiability of concepts, consistency checking of the KB as a whole, and various kinds of queries, including *subsumption* queries between concepts and database-like queries. These tasks are usually performed by *ontology reasoners*, often tailored towards specific applications and selection of DL languages.

One of the primary strengths of DLs resides in the ability to perform tasks taking into account both the explicit knowledge (ABox) and the implicit knowledge captured by the terminological representation of the domain (TBox). Different DL languages are designed to offer different levels of expressive power. However, expressiveness does not come without a computational cost; as such, when modelling a certain domain, particular attention needs to go into balancing the accuracy of the model with the complexity of the task that we want to perform on it.

This process of designing a KB by deciding on a relevant, domain-specific, vocabulary and building a model on top of it, is often referred to as *ontology engineering*. This process can be performed manually, semi-automatically, with the aid of visualization and building tools, or even automatically by means of learning procedures [8, 64, 96, 116]. *Ontology templates* [102, 103, 104] are also an active area of research and an effective way of guiding the process of designing and maintaining KBs. So far, this formalism has been applied to several domains, such as, astronomy [20], biology [93, 85], defence [63], education [17], energy management [16], medicine [15, 38, 47] and oil and gas [101, 58].

DLs also provide the logical underpinning for the Web Ontology Language (OWL) [40], cornerstone of the Semantic Web standard, as defined by the World Wide Web Consortium (W3C). As an example, the expressive DL language $\mathcal{SROIQ}$ [49] underpins OWL 2 DL [77], one of the more expressive languages in this family.

In the realm of data access, conjunctive query (CQ) answering is one of the primary reasoning tasks over KB for many applications. However, when considering expressive description logic languages, query answering is computationally very expensive, even when considering only complexity w.r.t. the size of the data (*data complexity*) [95, 32, 36, 25, 35, 82], and decidability is still an open problem when considering CQ answering over expressive languages like $\mathcal{SROIQ}$. Fully-fledged reasoners oriented towards CQ answering over OWL 2 ontologies exist but, although heavily optimized, often need to rely on restricting the expressive power of the input ontology or sacrifice completeness of the computed answers to achieve (empirical) tractability. This limiting process often targets particular constructs of DLs, such as the ability to represent *existential knowledge*

$$\mathtt{Lecturer} \sqsubseteq \exists \mathtt{teaches}.\mathtt{Course} \tag{1.1}$$

where we are claiming that every lecturer teaches at least one course, or *disjunctive knowledge*

$$\mathtt{Student} \sqsubseteq \mathtt{UndergradStudent} \sqcup \mathtt{GraduateStudent} \tag{1.2}$$

where we divide students by the level of degree they are currently pursuing.

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, 61, 68, 92, 107, 105].

An interesting fragment of OWL 2, tractable for standard reasoning tasks, is RSA. RSA [14] is an ontology language that extends the OWL 2 profiles, and for which a CQ answering algorithm, based on the *combined approach* technique [61, 60], was proposed by Feier, Carral, Stefanoni, et al. [29]. RSA is designed to avoid intractability due to *and-branching*, interaction between existential and universal knowledge that often leads to exponentially large models for a KB. By using a set of constraints on the structure of the KB, RSA is able to restrict itself to KBs with only polynomially bounded models.

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 that a tractable algorithm can 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 [120]. 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 like HermiT [31] 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. This was confirmed by our preliminary tests, as well. The computation of lower and upper bounds is achieved by under- and over-approximating the ontology into the OWL 2 profile OWL 2 RL so that a tractable reasoner can be used to answer the input queries. The tractability of OWL 2 RL is, again, achieved by avoiding problematic interactions between axioms that can cause an exponential blow-up of the computation. 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.

The objective of this research is to expand on this "pay-as-you-go" technique and improve existing CQ answering techniques over OWL 2 ontologies. We propose a new hybrid query answering architecture that combines black-box 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. The combined approach for RSA can then be used to compute the set of certain answers over the approximated ontology. Similarly, we propose an algorithm to compute an approximation "from above" targetting RSA$^+$; in this case, we consider an extension of RSA, enriched with additional axioms for the representation of (ir)reflexivity, asymmetry and disjointness among roles, aiming at computing an upper bound tighter than one computed by approximating to RSA. We prove that the combined approach for CQ answering for RSA is still complete when applied to an RSA$^+$ ontology.

These techniques led to the development of two new system: RSAComb and ACQuA (Answering CQs using Approximation)

RSAComb An efficient implementation [57, 56] of the combined approach algorithm for RSA [29], reorganized to fit the new implementation design and the integration of RDFox [79, 76, 74, 75] as a backend reasoner. We revise the overall structure of the combined approach for the language by improving some of the main steps and streamlining the execution of the algorithm by factoring out those tasks that are *query independent* to make answering multiple queries over the same knowledge base more efficient. 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 [53] of the hybrid architecture mentioned above, combining RSAComb, PAGOdA [120], and HermiT [31] 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, but tries to reduce the gap between upper and lower bounds by integrating approximations targetting more expressive language (i.e., RSA and RSA$^+$). Furthermore, we show how the additional computational cost introduced by reasoning over RSA can still provide a significant improvement compared to relying solely on a fully-fledged reasoner. The system has been designed 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 of the shortcomings of the latter tool. Our experimental results show that the new technique yields significant performance improvements in several important application scenarios. Both ACQuA[1] and RSAComb[2] have been released as free and open source software. Source code and documentation are available online.

This work is structured in two parts. Part I provides a summary of the logical foundations that are exploited in this thesis and an overview of the preliminary literature on description logics and conjunctive query answering. In particular

- Chapter 2 offers a recapitulation of the notation, syntax and semantics of FO logic, and its connection to rule-based knowledge representation languages;
- In Chapter 3, we introduce the $\mathcal{SROIQ}$ DL, along with associated standard reasoning tasks. We provide definitions and properties for different fragments of the language, such as the OWL 2 profiles and the RSA language;
- Chapter 4 focuses on CQ answering. We start by giving a formal definition of conjunctive query, along with a brief description of their representation as SPARQL queries. Following is an extensive analysis on the computational complexity of the problem, as well as a summary of the different answering techniques that can be encountered in literature.

Part II presents the hybrid approach to CQ answering implemented in ACQuA.

- In Chapter 5, we present the theoretical foundations behind the novel approach and focus in particular on the description of the novel approximation algorithms for the computation of answers bounds.

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

- Chapter 6 contains an in-depth description of the reference implementation of this hybrid system, ACQuA. We provide details on the overall structure and heuristics adopted during the implementation. Additional, technical, improvements to the implementation of the combined approach for RSA, in RSAComb, are also provided.

- Finally, Chapter 7 provide an extensive evaluation of performance and scalability of both RSAComb and ACQuA, and, in particular, a performance comparison of the latter with PAGOdA.

We conclude with a brief discussion on the contributions and present several opportunities for future work in Chapter 8.

Parts of this work have been previously published:

- The algorithm for the approximation of an unrestricted OWL 2 ontology to RSA, sound for CQ answering, was presented at ISWC 2021 [55].

- A full description of RSAComb system was presented at DL 2021 [57].

An extended version of the results presented in this thesis have been submitted to the Semantic Web Journal and are currently under review.

Part I

Foundations

2
Preliminaries

Contents

This chapter defines the preliminary notions needed in this thesis. We begin by giving an overview of syntax and semantics of First-Order (FO) logic, followed by its specialization into knowledge representation (KR) languages; we refer the reader to, e.g., [1], for a standard introduction to these topics. Throughout this work, we consider FO logic *without equality*, i.e., the standard equality predicate $\approx$ is treated as an ordinary predicate without any special semantics.

2.1 First-Order logic

2.1.1 Syntax

A *signature* Σ of a First-Order (FO) logic language is a tuple $\langle \mathcal{S}, \mathcal{V}, \mathcal{P}, \mathcal{F}, \mathcal{C} \rangle$ where

(a) $\mathcal{S} = \{\neg, \wedge, \vee, \rightarrow, \forall, \exists\}$ is a set of *reserved symbols* denoting negation, conjunction, disjunction, implication, universal and existential quantification, respectively.

(b) $\mathcal{V} = \{x, y, z, \dots\}$ is a set of *variables*.

(c) $\mathcal{P}$ is a set of *predicates*. We denote with $\mathcal{P}_n \subseteq \mathcal{P}$ the set of predicates of arity n for some $n \geq 0$. In particular, we consider special predicates $\bot, \top \in \mathcal{P}_0$.

(d) $\mathcal{F}$ is a set of *function symbols*. We denote with $\mathcal{F}_n \subseteq \mathcal{F}$ the set of function symbols of arity n for some $n \geq 1$.

(e) $\mathcal{C} = \{a, b, c, \dots\}$ is a set of *constants*.

We use $\vec{x}, \vec{y}, \vec{z}, \dots$ to denote vectors of variables, $|\vec{x}|$ to denote their arity and with slight abuse of notation we will occasionally interpret them as sets and write $x \in \vec{x}$ for any x occurring in $\vec{x}$. We recursively define *terms* as follows:

- any variable $x \in \mathcal{V}$ is a term;
- any constant $a \in \mathcal{C}$ is a term;
- given $t_1, \dots, t_n$ terms and $f \in \mathcal{F}_n$, then, $f(t_1, \dots, t_n)$ is a term.

An *atomic formula* (or simply *atom*) A is of the form $P(t_1, \dots, t_n)$, where $t_1, \dots, t_n$ are terms and $P \in \mathcal{P}_n$ a predicate of arity n. A *literal* is an atom A or its negation $\neg A$. We recursively define *formulas* as follows:

- any atomic formula A is a formula;
- if φ is a formula, then $\neg\varphi$ is a formula;
- if φ and ψ are formulas and $\circ \in \{\wedge, \vee, \rightarrow\}$ then, $\varphi \circ \psi$ is a formula;
- if φ is a formula and x is a variable, then, $\forall x \varphi$ and $\exists x \varphi$ are formulas.

We write $\forall \vec{x} \varphi$ (resp. $\exists \vec{x} \varphi$) to denote $\forall x_1 \dots \forall x_{|\vec{x}|} \varphi$ (reps. $\exists x_1 \dots \exists x_{|\vec{x}|} \varphi$). The *scope* of quantifiers and of *free* and *bound* variables in formulas can be defined recursively on the structure of the formula:

- Each variable occurring in an atom A is *free*.
- If x is free in φ, then it is free in $\neg\varphi$.
- For $\circ \in \{\wedge, \vee, \rightarrow\}$ and formulas φ and ψ, a variable x is free in $\varphi \circ \psi$ if it is free in either φ or ψ and bound otherwise.
- If x is free in φ, then it is free in $\forall \vec{y} \varphi$ (resp. $\exists \vec{y} \varphi$) if $x \notin \vec{y}$ and bound otherwise. Moreover, for each $y \in \vec{y}$ we say that y is in scope of $\forall$ (resp. $\exists$).

A *sentence* is a formula with no free variables. A *theory* is a set of sentences. W.l.o.g. we assume all bound variables in a formula are different and the set of bound and free variables are disjoint.

Given an atom A (resp. a formula φ), we denote with $terms(A)$ (resp. $terms(\varphi)$) the set of terms appearing in the atom (resp. formula). We call a term, an atom of a formula *ground* if they do not contain any variable. A *fact* is a function-free ground atom.

A *substitution* is a mapping from variables in a formula to terms. The expression $\sigma \equiv \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\} \equiv \{\vec{x} \mapsto \vec{t}\}$ denotes the mapping of variable x_i to term t_i for $1 \leq i \leq n$ and any other variable y to itself if $y \notin \vec{x}$. Moreover, we define $vars(\sigma) = \vec{x}$. A restriction of a mapping $\sigma = \{\vec{x} \mapsto \vec{t}\}$ to a set of variables $\vec{y} = \{y_1, \ldots, y_m\}$ is denoted by $\sigma|_{\vec{y}} = \{y_1 \mapsto \sigma(y_1), \ldots, y_m \mapsto \sigma(y_m)\}$. The application of a substitution $\sigma = \{\vec{x} \mapsto \vec{t}\}$ to a term t, written $t\sigma$ or $t|_{\vec{x}\mapsto\vec{t}}$ can be recursively defined as

- $c\sigma = c$ for every constant $c \in \mathcal{C}$,
- $x\sigma = \sigma(x)$ for every variable $x \in \mathcal{V}$,
- $(f(t_1, \ldots, t_n))\sigma = f(t_1\sigma, \ldots, t_n\sigma)$ for $t_1, \ldots, t_n$ terms and $f \in \mathcal{F}_n$.

The application of a substitution σ to a formula φ, written $\varphi\sigma$ or $\varphi|_{\vec{x}\mapsto\vec{t}}$, can be recursively defined as

- $(P(t_1, \ldots, t_n))\sigma = P(t_1\sigma, \ldots, t_n\sigma)$ for $t_1, \ldots, t_n$ terms and $P \in \mathcal{P}_n$,
- $(\neg\varphi)\sigma = \neg(\varphi\sigma)$,
- $(\varphi \circ \psi) = \varphi\sigma \circ \psi\sigma$ for $\circ \in \{\wedge, \vee, \rightarrow\}$,
- $(Q\vec{y}\varphi)\sigma = Q\vec{y}(\varphi\sigma|_{vars(\sigma)\setminus\vec{y}})$ for $Q \in \{\exists, \forall\}$.

Substitution is extended to other concepts introduced above in a straightforward way.

Finally, let Σ be a FO signature that includes the equality symbol $\approx$.[1] Then we define the *equality axiomatization* of Σ as the set of FO sentences

$$\forall x(x \approx x) \tag{2.1}$$

$$\forall x \forall y(x \approx y \rightarrow y \approx x) \tag{2.2}$$

$$\forall x \forall y \forall z(x \approx y \wedge y \approx z \rightarrow x \approx z) \tag{2.3}$$

$$\forall \vec{x} \forall x_i'(x_i \approx x_i' \wedge P(\vec{x}) \rightarrow P(x_1, \ldots, x_i', \ldots, x_n)) \tag{2.4}$$

$$\forall \vec{x} \forall x_i'(x_i \approx x_i' \rightarrow f(\vec{x}) \approx f(x_1, \ldots, x_i', \ldots, x_n)) \tag{2.5}$$

[1] W.l.o.g. we will write $x \approx y$ instead of $\approx(x, y)$

where $\vec{x} = \langle x_1, \ldots, x_i, \ldots, x_n \rangle$ and $1 \leq i \leq n$. Formulas (2.1), (2.2) and (2.3) denote *reflexivity*, *symmetry*, and *transitivity*, and formula (2.4) (resp. formula (2.5)) denotes the *substitution rule* for every $P \in \mathcal{P}_n$ (resp. for every $f \in \mathcal{F}_n$) and for every arity n.

2.1.2 Semantics

An *interpretation* is a tuple $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ consisting of a non-empty domain set $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ defined as follows:

- every constant $c \in \mathcal{C}$ is mapped to an element $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$,
- every function $f \in \mathcal{F}_n$, for any arity n, is mapped to $f^{\mathcal{I}} : (\Delta^{\mathcal{I}})^n \to \Delta^{\mathcal{I}}$
- every predicate $P \in \mathcal{P}_n$, for any arity n, is mapped to $P^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$

where we denote with $(\Delta^{\mathcal{I}})^n$ the set of n–tuples with elements in $\Delta^{\mathcal{I}}$. Given t a ground term, $t^{\mathcal{I}}$ is recursively defined as follows:

- if $t = c$ with $c \in \mathcal{C}$, then $t^{\mathcal{I}} = c^{\mathcal{I}}$,
- if $t = f(t_1, \ldots, t_n)$ for $f \in \mathcal{F}_n$ and $t_1, \ldots, t_n$ terms, then $t^{\mathcal{I}} = f^{\mathcal{I}}(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}})$.

Given φ a FO sentence, we use $\varphi^{\mathcal{I}}$ to denote the *truth value* of φ w.r.t. the interpretation $\mathcal{I}$.

- For any $P \in \mathcal{P}_n$ with $n > 0$ and $t_1, \ldots, t_n$ terms.

$$(P(t_1, \ldots, t_n))^{\mathcal{I}} = \begin{cases} \textsc{True} & \text{if } \langle t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}} \rangle \in P^{\mathcal{I}} \\ \textsc{False} & \text{if } \langle t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}} \rangle \notin P^{\mathcal{I}} \end{cases} \tag{2.6}$$

- $(\neg \varphi)^{\mathcal{I}} = \neg(\varphi^{\mathcal{I}})$
- For $\circ \in \{\wedge, \vee, \to\}$, $(\varphi \circ \psi)^{\mathcal{I}} = \varphi^{\mathcal{I}} \circ \psi^{\mathcal{I}}$
- $(\forall x \varphi)^{\mathcal{I}} = \textsc{True}$ if $\varphi|_{x \mapsto d} = \textsc{True}$ for all $d \in \Delta^{\mathcal{I}}$
- $(\exists x \varphi)^{\mathcal{I}} = \textsc{True}$ if $\varphi|_{x \mapsto d} = \textsc{True}$ for some $d \in \Delta^{\mathcal{I}}$

Let $\mathcal{I}$ be an interpretation, and let φ be a FO sentence. We say that $\mathcal{I}$ *models* (or *satisfies*) φ, written $\mathcal{I} \models \varphi$, if $\varphi^{\mathcal{I}}$ is True. Alternatively, we say that $\mathcal{I}$ is a *model* for φ. Given a theory Φ, $\mathcal{I} \models \Phi$ if $\mathcal{I} \models \varphi$, for every $\varphi \in \Phi$. We say that Φ is *satisfiable* if there exists an interpretation satisfying it. A theory Φ satisfies a FO sentence ψ (written $\Phi \models \psi$) if every model for Φ is a model for ψ.

2.1.3 Herbrand interpretations

Let us consider a fixed FO signature Σ. Then, we call *Herbrand universe* $\mathcal{H}_U$ the set of all ground terms and *Herbrand base* $\mathcal{H}_B$ the set of all ground atoms. Given $M \subseteq \mathcal{H}_B$ such that $\bot \notin M$, the *Herbrand interpretation* $\mathcal{I}_M = \langle \Delta^{\mathcal{I}_M}, \cdot^{\mathcal{I}_M} \rangle$ is defined as

- $\Delta^{\mathcal{I}_M} = \mathcal{H}_U$,
- $c^{\mathcal{I}_M} = c$ for $c \in \mathcal{C}$,
- $f^{\mathcal{I}_M} = f$ such that for any term $t = f(t_1, \ldots, t_n)$, $f \in \mathcal{F}_n$ and some n, $t^{\mathcal{I}} = f(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}})$,
- $P^{\mathcal{I}_M} = \{\langle t_1, \ldots, t_n \rangle \mid P(t_1, \ldots, t_n) \in M\}$ for each $P \in \mathcal{P}_n$ for some arity n.

Let Φ be a theory over Σ. A Herbrand interpretation $\mathcal{I}_M$ over Σ is a *Herbrand model* of Φ if $\mathcal{I}_M \models \Phi$. Let M and N be two sets of ground atoms. A *homomorphism* from M to N is a mapping τ from ground terms in M to ground terms in N such that for any $A \in M$, $A\tau \in N$. A Herbrand model $\mathcal{I}_M$ for a FO theory Φ is a *universal model* for Φ if, for any Herbrand model $\mathcal{I}_N$ for Φ, there exists a homomorphism from M to N.

2.2 Rule-based knowledge representation

Given a FO signature Σ, we define a *rule* as a FO sentence of the form

$$\forall \vec{x} \forall \vec{y} (\beta_1(\vec{x}, \vec{y}) \land \cdots \land \beta_n(\vec{x}, \vec{y}) \to \bigvee_{i=1}^{m} \exists \vec{z}_i \varphi_i(\vec{x}, \vec{z}_i)) \tag{2.7}$$

where $\vec{x}, \vec{y}, \vec{z}_i$ are pair-wise disjoint sets of variables, $\beta_i(\vec{x}, \vec{y})$ literals with variables in $\vec{x} \cup \vec{y}$ and either

- $m = 1$, and $\varphi_1(\vec{x}, \vec{z}_1) = \bot$, or
- $m \geq 1$, and $\varphi_i(\vec{x}, \vec{z}_i)$ is a conjunction of atoms with variables in $\vec{x} \cup \vec{z}_i$.

The conjunction of literals $\beta_1(\vec{x}, \vec{y}) \land \cdots \land \beta_n(\vec{x}, \vec{y})$ is the *body* of r, denoted as $\texttt{body}(r)$. Moreover, we denote with $\texttt{body}^+(r)$ the set of positive atoms in the body of r and with $\texttt{body}^-(r)$ the set of negative atoms in the body of r. The formula $\bigvee_{i=1}^{m} \exists \vec{z}_i \varphi_i(\vec{x}, \vec{z}_i)$ is the *head* of r, denoted as $\texttt{head}(r)$. A rule r is *safe* if all variables in $\vec{x}$ occur in $\texttt{body}^+(r)$. We consider only safe rules.

We can classify a rule r as

- *Horn* if $m = 1$;
- *definite* if Horn, $\vec{z}_i$ is empty for every i, and $\texttt{body}^-(r)$ is empty;
- *disjunctive Datalog* if function-free, $\vec{z}_i$ is empty for every $1 \leq i \leq m$, and $\texttt{body}^-(r)$ is empty;
- *Datalog* if both disjunctive Datalog and Horn (or alternatively if both definite and function-free).

By slight abuse of notation, we consider facts as definite rules with an empty ($\top$) body and, if r is Horn, we consider $\texttt{head}(r)$ to be the set of conjuncts in the head of the rule. A *program* is a finite set of rules. The concepts of Horn, definite, disjunctive Datalog and Datalog can be extended to programs in a straightforward way.

2.2.1 Program stratification

Given $\texttt{preds}(\cdot)$ the function that returns the set of predicates in $\mathcal{P}$ in either a formula, a set of atoms or a program, a *stratification* for a program Π is a function $\delta : \texttt{preds}(\Pi) \rightarrow \{1, \ldots, k\}$ where $k \leq |\texttt{preds}(\Pi)|$ and such that, for every $r \in \Pi$ and $P \in \texttt{preds}(\texttt{head}(r))$

- for every $Q \in \texttt{preds}(\texttt{body}^+(r))$ then, $\delta(Q) \leq \delta(P)$
- for every $Q \in \texttt{preds}(\texttt{body}^-(r))$ then, $\delta(Q) < \delta(P)$

The *stratification partition* of Π induced by δ is the sequence $(\Pi_1, \ldots, \Pi_k)$, with each Π_i, also called *stratum*, consisting of all rules $r \in \Pi$ such that

$$\texttt{max}_{P \in \texttt{preds}(\texttt{head}(r))}(\delta(P)) = i \tag{2.8}$$

A program is *stratified* if it admits a stratification. According to this definition, all definite programs are stratified.

Every stratified program Π has a least Herbrand model (LHM), i.e., a unique Herbrand model, minimal w.r.t. set inclusion, that can be constructed using the immediate consequence operator T_Π.

Let $S \subseteq \mathcal{H}_B$, 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

- $T^0_\Pi(S) = S$
- $T^{n+i}_\Pi(S) = T_\Pi(T^n_\Pi(S))$
- $T^\omega_\Pi(S) = \bigcup_{i=0}^{\infty} T^i_\Pi(S)$

Let $(\Pi_1, \ldots, \Pi_k)$ be a stratification partition for Π. We define $U_1 = T^\omega_{\Pi_1}(\emptyset)$ and for each $1 \leq i < k$, $U_{i+1} = T^\omega_{\Pi_{i+1}}(U_i)$. Then, the LHM of Π is U_k and is denoted as $M[\Pi]$.

Finally, for each $P \in \mathcal{P}$ for some signature Σ, we consider the rule

$$P(x_1, \ldots, x_n) \to \top(x_1), \ldots, \top(x_n) \tag{2.9}$$

Given a program Π, we denote with $\Pi^{\approx,\top}$ the extension of Π with *top axiomatization* rules (2.9) and *equality axiomatization* rules (2.1) to (2.5).

2.2.2 Rule Skolemization

In various occasions throughout this thesis we will refer to the concept of *Skolemization*. Given a rule r of the form (2.7), for each existential quantifier variable z_{ij}, let f^r_{ij} be a function symbol globally unique for r and z_{ij} of arity $|\vec{x}|$. Furthermore, let $\sigma_{\mathtt{sk}}$ be the substitution such that $\sigma_{\mathtt{sk}}(z_{ij}) = f^r_{ij}(\vec{x})$ for each $z_{ij} \in \vec{z}_i$. The Skolemization of r is the following FO sentence

$$\forall \vec{x} \forall \vec{y} (\beta_1(\vec{x}, \vec{y}), \ldots, \beta_n(\vec{x}, \vec{y}) \to \bigvee_{i=1}^{m} \varphi_i(\vec{x}, \vec{z}_i)\sigma_{\mathtt{sk}}) \tag{2.10}$$

Now, for each existential quantifier variable z_{ij}, let c^r_{ij} be a fresh constant symbol globally unique for r and z_{ij}. Furthermore, let $\sigma_{\mathtt{csk}}$ be the substitution such that $\sigma_{\mathtt{csk}}(z_{ij}) = c^r_{ij}$ for each $z_{ij} \in \vec{z}_i$. The *constant* Skolemization of r is the following FO sentence

$$\forall \vec{x} \forall \vec{y} (\beta_1(\vec{x}, \vec{y}), \ldots, \beta_n(\vec{x}, \vec{y}) \to \bigvee_{i=1}^{m} \varphi_i(\vec{x}, \vec{z}_i)\sigma_{\mathtt{csk}}) \tag{2.11}$$

The (constant) Skolemization operator can be trivially extended to programs.

It is well-known that Skolemization is an entailment-preserving transformation, i.e., for an arbitrary program Π and a FO sentence φ in the signature of Π, $\Pi \models \varphi$ iff $\Pi\sigma_{\mathtt{sk}} \models \varphi$. This is not the case for constant Skolemization.

3
Description logics

Contents

In this chapter we introduce the syntax and semantics of the DLs underpinning the Web Ontology Language (OWL) and its variants and fragments. We will start by introducing syntax and semantics of $\mathcal{SROIQ}$ [49, 119], the logical underpinning of OWL 2 DL, along with a normal form for the language and its translation into logic rules. Finally, we will provide additional details for the OWL 2 profiles, standardized fragments of OWL 2 with particularly valuable properties.

3.1 The description logic $\mathcal{SROIQ}$

A description logic (DL) language defines a syntax to build axioms, i.e., first order sentences, and assertions, i.e., atoms, both restricted to unary and binary predicates.

A	(concept name)
$\{a\}$	(nominal)
$C \sqcap D$	(conjunction)
$C \sqcup D$	(disjunction)
$\neg C$	(negation)
$\forall R.C$	(value restriction)
$\exists R.C$	(existential restriction)
$\exists R.\mathtt{Self}$	(self restriction)
$\leq nR.C$	(max number restriction)
$\geq nR.C$	(min number restriction)

Table 3.1: Concepts in $\mathcal{SROIQ}$, with $A \in N_C$, $a \in N_I$, $n \in \mathbb{N}$, R role and C, D concepts.

3.1.1 Syntax

A signature for $\mathcal{SROIQ}$ consists of the following symbols and operators:

- $\neg$, $\sqcap$, $\sqcup$, $\sqsubseteq$ (negation, conjunction, disjunction, and implication),
- $\forall$, $\exists$, $\leq$, $\geq$ (universal and existential quantification, min/max cardinality),
- $^-$, $\circ$, `Self` (role inverse, composition and self constructor),
- `Ref` (reflexivity), `Irr` (irreflexivity), `Sym` (symmetry), `Asy` (asymmetry), `Trans` (transitivity), `Dis` (disjointness), `Func` (functionality),

and pair-wise, disjoint, countable sets N_C, N_R, N_I of unary predicates, binary predicates and constants (*individuals*), respectively. Predicates in N_C are called *concept names* and predicates in N_R are called *role names*. We assume $\{\bot, \top\} \in N_C$. The set of *roles* is defined as $N_R \cup \{R^- \mid R \in N_R\}$ where R^- is the *inverse role* of R. We define $\mathtt{Inv}(\cdot)$ as

$$\mathtt{Inv}(R) = \begin{cases} R^- & \text{if } R \in N_R \\ S & \text{if } R \equiv S^- \text{ with } S \in N_R \end{cases}$$

Concepts[1] are defined inductively according to Table 3.1. An *axiom* is either

- a *general concept inclusion (GCI)*, of the form $C \sqsubseteq D$ with C,D concepts. We use $C \equiv D$ to abbreviate the axioms $C \sqsubseteq D$ and $D \sqsubseteq C$;
- a *role axiom* of the forms $\mathtt{Ref}(R)$, $\mathtt{Irr}(R)$, $\mathtt{Sym}(R)$, $\mathtt{Asy}(R)$, $\mathtt{Trans}(R)$, $\mathtt{Dis}(R)$, or $R_1 \circ \cdots \circ R_n \sqsubseteq R$, with $R, S, R_1, \ldots, R_n$ roles and the last form called *role inclusion.* A role inclusion axiom is *complex* if $n > 1$.

[1]The DL terms “concept” and “role” correspond to the OWL terms “class” and “property” in the W3C standards.

	Axioms α	Logic rules $\pi(\alpha)$
(R1)	R^-	$R(x,y) \to R^-(y,x) \qquad R^-(y,x) \to R(x,y)$
(R2)	$R \sqsubseteq S$	$R(x,y) \to S(x,y)$
(R3)	$R \sqcap S \sqsubseteq \bot$	$R(x,y) \wedge S(x,y) \to \bot$
(R4)	$R \circ S \sqsubseteq T$	$R(x,y) \wedge S(y,z) \to T(x,z)$
(T1)	$\bigsqcap_{i=1}^n A_i \sqsubseteq \bigsqcup_{i=1}^m B_i$	$\bigwedge_{i=1}^n A_i(x) \to \bigvee_{i=1}^m B_i(x)$
(T2)	$A \sqsubseteq \{a\}$	$A(x) \to x \approx a$
(T3)	$\exists R.A \sqsubseteq B$	$R(x,y) \wedge A(y) \to B(x)$
(T4)	$A \sqsubseteq \leq mR.B$	$A(x) \wedge \bigwedge_{i=1}^{m+1}[R(x,y_i) \wedge B(y_i)] \to \bigvee_{1 \leq i < j \leq m+1} y_i \approx y_j$
(T5)	$A \sqsubseteq \exists R.B$	$A(x) \to R(x, f_{R,B}^A(x)) \wedge B(f_{R,B}^A(x))$
(T6)	$A \sqsubseteq \mathtt{Self}(R)$	$A(x) \to R(x,x)$
(T7)	$\mathtt{Self}(R) \sqsubseteq A$	$R(x,x) \to A(x)$
(A1)	$A(a)$	$\to A(a)$
(A2)	$R(a,b)$	$\to R(a,b)$

Table 3.2: Normalized $\mathcal{SROIQ}$ axioms and their translation into logic rules.

A *concept assertion* is a ground atom of the form $C(a)$ with C a concept and $a \in N_I$. A *role assertion* is a ground atom of the form $R(a,b)$ with R role and $a, b \in N_I$.

According to the DL naming convention (see Appendix B), $\mathcal{SROIQ}$ can be defined as the basic DL $\mathcal{S}$ with the addition of complex role composition ($\mathcal{R}$), nominals ($\mathcal{O}$), inverse roles ($\mathcal{I}$) and qualified number restriction ($\mathcal{Q}$). Note that, in this particular context, $\mathcal{SROIQ}$ also includes disjointness, (ir)reflexivity, (a)symmetry axioms as well as self restriction, even though this is not reflected by the presence of the relevant symbols.

Table 3.2 introduces a normal form for $\mathcal{SROIQ}$ [119]. W.l.o.g. we assume that any axiom defined above can be converted into a set of axioms in Table 3.2, and, as such, the following definitions are based on this syntax.

A role is *complex* if it is a conjunction of roles ($R \sqcap S$), or composition ($R \circ S$) of roles. An *RBox* $\mathcal{R}$ is a finite set of axioms of type (R2)–(R4) with R, S, T roles. We denote $\sqsubseteq_{\mathcal{R}}^*$ as the 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 *TBox* $\mathcal{T}$ is a set of axioms of type (T1)–(T7) where $A, B \in N_C$, $a \in N_I$ and R 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 $\mathcal{O} = \mathcal{T} \cup \mathcal{R}$. 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 forcing $m = 1$ in axioms (T1) and (T4). Finally, given an ontology language $\mathcal{L}$, we define an $\mathcal{L}$ *knowledge base (KB)* as a tuple $\langle \mathcal{O}, \mathcal{A} \rangle$ comprising an $\mathcal{L}$ ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{R}$ and an ABox $\mathcal{A}$.[2]

Each normalized axiom corresponds to a logic rule, as given on the right hand-side of Table 3.2. We call $\pi(\cdot)$ the function that converts normalized axioms and assertions into rules; given $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$, we denote the program $\pi(\mathcal{K}) = \{\pi(\alpha) \mid \alpha \in \mathcal{O} \cup \mathcal{A}\}$ and $\pi(\mathcal{K})^{\approx,\top}$ as the smallest set containing all rules in $\pi(\mathcal{K})$ and axiomatization rules for equality ($\approx$) and $\top$ as defined in Section 2.1.1.

Syntactical restrictions in $\mathcal{SROIQ}$

In order to ensure decidability of several reasoning tasks (see Section 3.2), the definition of $\mathcal{SROIQ}$ involves a number of additional syntactical restrictions. Given a $\mathcal{SROIQ}$ ontology $\mathcal{O}$, a *simple role* R is inductively defined as follows:

i) R is a role name which does not appear on the right hand-side of any role inclusion in $\mathcal{O}$,

ii) R is the inverse of a simple role,

iii) R only appears in role inclusions of the form $S \sqsubseteq R$ with S simple.

In $\mathcal{SROIQ}$, axioms of type (R3), (T4)–(T7) are restricted to simple roles.

Moreover, let $\prec$ be a partial ordering over roles, then $\prec$ is a regular order if $S \prec R \Leftrightarrow \texttt{Inv}(S) \prec R$, for every R,S roles. An RBox $\mathcal{R}$ is *regular* if and only if there exists a regular partial order $\prec$ over $\mathcal{R}$ and every axiom in $\mathcal{R}$ is of the form

i) $R \circ R \sqsubseteq R$,

ii) $\texttt{Inv}(R) \sqsubseteq R$,

iii) $R \circ R_1 \circ \cdots \circ R_n \sqsubseteq R$,

iv) $R_1 \circ \cdots \circ R_n \circ R \sqsubseteq R$;

where $R_i \prec R$, for every $1 \leq i \leq n$, whenever R_i is *not* a simple role. In a $\mathcal{SROIQ}$ ontology, the RBox is required to be regular.

[2]Sometimes we say that an axiom α is part of a KB $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$ (in symbols, $\alpha \in \mathcal{K}$), to denote that $\alpha \in \mathcal{O}$

Syntax	Semantics
$\top$	$\Delta^{\mathcal{I}}$
$\bot$	$\emptyset$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
$\leq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \in \Delta^{\mathcal{I}} : \langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$
$\texttt{Self}(R)$	$\{x \in \Delta^{\mathcal{I}} \mid \langle x, x\rangle \in R^{\mathcal{I}}\}$
R^{-}	$\{\langle y, x\rangle \mid \langle x, y\rangle \in R^{\mathcal{I}}\}$
$R \sqcap S$	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$
$R \circ S$	$\{\langle a, c\rangle \mid \langle a, b\rangle \in R^{\mathcal{I}}, \langle b, c\rangle \in S^{\mathcal{I}}\}$

Table 3.3: Extension of interpretation function $\cdot^{\mathcal{I}}$ to $\mathcal{SROIQ}$ concepts.

3.1.2 Semantics

Given a $\mathcal{SROIQ}$ signature Σ, an interpretation $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}\rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set called *domain* and $\cdot^{\mathcal{I}}$ is an *interpretation function* defined for each element of N_C, N_R, N_I:

- for each concept name $C \in N_C$, $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
- for each role name $R \in N_R$, $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$,
- for each individual name $a \in N_I$, $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

The interpretation function definition can be extended to concepts and complex roles as described in Table 3.3.

Furthermore, the interpretation function defines the satisfaction condition for axioms and assertions:

- for every concept assertion $C(a)$, $\mathcal{I} \models C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$,
- for every role assertion $R(a, b)$, $\mathcal{I} \models R(a, b)$ iff $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$,
- for every $C \sqsubseteq D$, with C, D concepts, $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$,
- for every $S \sqsubseteq R$, with S complex role and R role, $\mathcal{I} \models S \sqsubseteq R$ iff $S^{\mathcal{I}} \subseteq R^{\mathcal{I}}$.

As for the FO case, we say that an interpretation $\mathcal{I}$ *models* (is a *model* for, *satisfies*) an axiom (resp. assertion) α iff $\mathcal{I} \models \alpha$. Given an ontology $\mathcal{O}$, $\mathcal{I} \models \mathcal{O}$ if $\mathcal{I} \models \alpha$, for every $\alpha \in \mathcal{O}$. Additionally, given a KB $\mathcal{K} = \langle \mathcal{O}, \mathcal{A}\rangle$, $\mathcal{I} \models \mathcal{K}$ if $\mathcal{I} \models \mathcal{O}$ and $\mathcal{I} \models \alpha$, for every $\alpha \in \mathcal{A}$. We say that $\mathcal{K}$ is *satisfiable* if there exists an interpretation satisfying it. A KB $\mathcal{K}$ satisfies an axiom (or an assertion) α (written $\mathcal{K} \models \alpha$) if every model for $\mathcal{K}$ is a model for α.

Given $a \in N_I$ and a concept C, such that $a^{\mathcal{I}} \in C^{\mathcal{I}}$, for some interpretation $\mathcal{I}$, then, we say that a is an *instance* of C w.r.t. $\mathcal{I}$.

Finally, the translation $\pi(\cdot)$ from ontology axioms to rules, defined in the previous section is *entailment preserving*, i.e., $\mathcal{K} \models \alpha$ iff $\pi(\mathcal{K})^{\approx,\top} \models \pi(\alpha)$ for any axiom or assertion α, and as such we can treat the two formalisms as interchangeable.

The semantics for $\mathcal{SROIQ}$ can be equivalently defined via translation of axioms into FO formulas and by referring to FO logic semantics, as defined in Section 2.1.2. The translation into logic rules is given in Table 3.2, with (T5) axioms translated to $A(x) \rightarrow \exists y(R(x,y) \wedge B(y))$.

3.2 Reasoning problems

Following is a list of standard reasoning problems that are associated with DLs. Given $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$ a KB with signature $\Sigma = \langle N_C, N_R, N_I \rangle$, $\mathcal{I}$ an interpretation for $\mathcal{K}$, A, B concept names, C, D concepts:

Consistency checking (or *satisfiability checking*) is the problem of deciding whether there exists an interpretation $\mathcal{I}$ that satisfies $\mathcal{K}$.

Subsumption is the problem of deciding whether each instance of C is also an instance of D in all models of $\mathcal{K}$.

Classification is the task of determining the subsumption relation for all pairs of concept names A,B in $\mathcal{K}$.

Instance Retrieval is the task of collecting all individuals $a \in N_I$ such that a is an instance of C in all models of $\mathcal{K}$.

Realization is the task of computing, for every individual $a \in N_I$ the minimal set of concept names $A \in N_C$ such that a is an instance of A for all models of $\mathcal{K}$.

3.3 OWL 2 profiles

OWL 2 *profiles* [40, 72] have been defined as fragments of OWL 2, and designed to provide a desirable balance between computational complexity of standard reasoning tasks and expressiveness of the ontology language. OWL 2 provides three profiles: OWL 2 EL, OWL 2 QL and OWL 2 RL.

3.3.1 OWL 2 EL

OWL 2 EL is based on the $\mathcal{EL}^{++}$ family of DLs, and has been designed to allow for efficient reasoning over large ontologies. The main reasoning task of interest is classification [40], and standard reasoning (see Section 3.2) in this profile can be implemented in polynomial time w.r.t. the size of the ontology [4].

In OWL 2 EL, concepts are formed according to the following:

$$C, D \rightarrow \bot \mid A \mid \{a\} \mid C \sqcap D \mid \exists R.C \mid \exists R.\mathtt{Self} \tag{3.1}$$

where A is a concept name, C, D concepts, and R is a role name. A GCI in OWL 2 EL is of the form $C \sqsubseteq D$ with C, D concepts; $\mathtt{Ref}(R)$, $\mathtt{Trans}(R)$ and complex role inclusion are also allowed.

3.3.2 OWL 2 QL

OWL 2 QL is based on the DL-Lite family [10] of DLs, which has been tailored towards efficient reasoning over large amounts of data enriched by a relatively simple ontology schema. The main reasoning task for OWL 2 QL is conjunctive query (CQ) answering (see Section 4.1). This task is usually implemented by means of query rewriting techniques (see Section 4.4.3) where a CQ is rewritten into a union of CQs that captures the information introduced by the ontology; the rewritten query can be answered over the input dataset using conventional RDBMSs. OWL 2 QL ensures that a polynomial rewriting of a query exists [10].

OWL 2 QL is based on the DL-Lite$_{\mathcal{R}}$ DL; concepts are defined as

$$C \rightarrow \bot \mid A \mid \exists R \tag{3.2}$$

where A is a concept name, and R a role. A GCI in OWL 2 QL is of the form $C \sqsubseteq D$ with C, D concepts. Role axioms are restricted to $\mathtt{Ref}(R)$, $\mathtt{Sym}(R)$, $\mathtt{Asy}(R)$, $\mathtt{Dis}(R, S)$ with R, S roles and simple concept inclusion.

3.3.3 OWL 2 RL

OWL 2 RL is based on the Description Logic Program (DLP) [41] formalism, placing itself in the intersection between DLs and Datalog. CQ answering in OWL 2 RL is P–complete in data complexity [1].

GCIs $C \sqsubseteq D$ in OWL 2 RL are defined as follows

$$C \rightarrow A \mid \{a\} \mid C \sqcap C \mid \exists R.C \tag{3.3}$$

$$D \rightarrow \bot \mid A \mid \neg C \mid \forall R.D \mid \exists R.\{a\} \mid \leq 1 R.C \tag{3.4}$$

where A is a concept name and R a role.

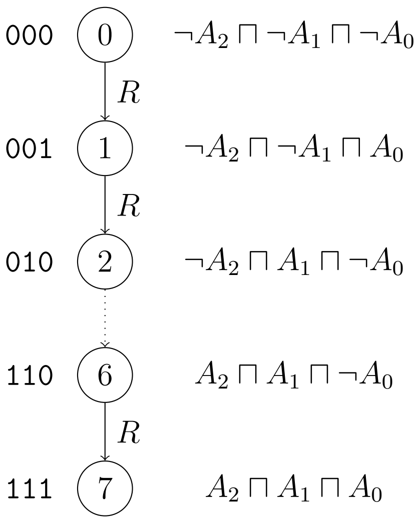


Figure 3.1: Example of exponential model enumerating numbers from 0 to $2^n - 1$ for $n = 3$. The KB is polynomial in n.

3.4 RSA

Role safety acyclic (RSA) ontologies were originally presented by Carral, Feier, Grau, et al. [14] and further analyzed by Feier, Carral, Stefanoni, et al. [29].

RSA is a class of ontology languages designed to subsume all OWL 2 profiles, while maintaining tractability of the 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* [4] and can be caused by interactions between axioms of type (T5) with either axioms (T3) and (R1), or axioms (T4), in Table 3.2.

Example 3.4.1. Interaction between existential quantifiers (which can be encoded as axioms of type (T5)) and universal quantifiers (which can be encoded by axioms of type (T3) and (R2)) 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 $0 \leq i < n$:

$$\neg A_i \sqcap \bigsqcap_{j<i} A_j \sqsubseteq B_i \sqcap \exists R.A_i \sqcap \forall R.(\bigsqcap_{j<i} \neg A_j) \tag{3.5}$$

$$\forall_{j>i}\Big(B_i \sqcap A_j \sqsubseteq \forall R.A_j\Big) \tag{3.6}$$

$$\forall_{j>i}\Big(B_i \sqcap \neg A_j \sqsubseteq \forall R.\neg A_j\Big) \tag{3.7}$$

The knowledge base, of polynomial size w.r.t. n, enforces a chain of individuals 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 3.1 shows an example of the exponentially large model induced by the TBox 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-$\mathcal{ALCHOIQ}$ KB $\mathcal{K} = \langle \mathcal{T} \cup \mathcal{R}, \mathcal{A} \rangle$ over the signature $\Sigma_{\mathcal{K}} = \langle N_C, N_R, N_I \rangle$.

Definition 3.4.1. *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. *$R \sqsubseteq^*_{\mathcal{R}} Inv(S)$ 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 $R \sqsubseteq^*_{\mathcal{R}} Inv(S)$.*

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

Note that, all OWL 2 profiles, as defined in Section 3.3, contain only safe roles. The distinction between safe and unsafe roles makes it possible to strengthen the translation π from Table 3.2 as follows:

Definition 3.4.2. *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 $\mathcal{K}$:*

$$\pi_{safe}(\alpha) = \begin{cases} A(x) \to R(x, v^A_{R,B}) \wedge B(v^A_{R,B}) & \textit{if } \alpha \textit{ of type } \text{(T5)} \textit{ and } R \textit{ safe} \\ \pi(\alpha) & \textit{otherwise.} \end{cases} \tag{3.8}$$

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

Theorem 3.4.1 ([14], Theorem 2)**.** *A Horn-$\mathcal{ALCHOIQ}$ KB $\mathcal{K}$ 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 $M[\mathcal{P}_{\mathcal{O}}]$ might be infinite (or exponentially large), due to the introduction of function symbols caused by the Skolemization of existential axioms.

Definition 3.4.3. *Let* $\mathtt{PE}$ *and* $\mathtt{E}$ *be fresh binary predicates, let* $\mathtt{U}$ *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* $\mathcal{K}$

$$\pi_{RSA}(\alpha) = \begin{cases} A(x) \to R(x, u^A_{R,B}) \wedge B(u^A_{R,B}) \wedge \mathtt{PE}(x, u^A_{R,B}) & \text{if } \alpha \text{ is of type (T5)} \\ \pi(\alpha) & \text{otherwise.} \end{cases} \tag{3.9}$$

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

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

(i) for each pair of atoms $w \approx t$ *(with* w *and* t *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. If the KB* $\mathcal{K}$ *is Horn-*$\mathcal{ALCHOIQ}^+$*,* equality-safe *and* $G_\mathcal{K}$ *is an oriented forest, we say that* $\mathcal{K}$ *is RSA*$^+$*.*

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-$\mathcal{ALCHOIQ}^+$) ontology is RSA (resp. RSA$^+$).

Theorem 3.4.2 ([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 3.4.1 and Theorem 3.4.2.

Finally, we introduce an alternative definition of OWL 2 profiles as fragments of Horn-$\mathcal{ALCHOIQ}$. Unless otherwise stated, we will use this definition of OWL 2 profiles, which does not consider property chain and transitivity axioms, in order to keep this work compatible with definitions by Feier, Carral, Stefanoni, et al. [29].

OWL 2 profiles can be defined as restrictions of Horn-$\mathcal{ALCHOIQ}$ as follows:

- OWL 2 EL does not allow inverse roles (R1) and axioms of type (T4),
- OWL 2 RL does not allow axioms of type (T5), and

- OWL 2 QL does not allow axioms (T2), (T4), axioms (T1) satisfy $n = 1$ and axioms (T3) satisfy $A = \top$.

It is worth noting that, when not considering *transitive roles*, the logical underpinning of OWL 2 EL matches $\mathcal{ELHO}^r_\bot$ [105, 120].

4

Query answering over ontologies

Contents

Conjunctive query (CQ) answering over ontologies is a reasoning task that has received increasing attention in recent years. In this chapter we introduce the definition of *conjunctive query* and its answers w.r.t. a knowledge base [4, 84, 119]. Computational complexity of CQ answering has been thoroughly investigated in the literature and a number of different algorithms to compute all answers to a query have been proposed.

4.1 Conjunctive queries

A *conjunctive query (CQ)* is a FO formula $q(\vec{x}) = \exists\vec{y}\psi(\vec{x},\vec{y})$ where $\psi(\vec{x},\vec{y})$ is a conjunction of function-free atoms over $\vec{x} \cup \vec{y}$, $\vec{y}$ are called *existential* or *bound variables* and $\vec{x}$ are called *answer variables*. For the sake of simplicity, we sometimes omit the existential variables from the query and write q instead of $q(\vec{x})$. W.l.o.g. we treat a query as the set of its conjuncts. A CQ $q(\vec{x}) = \exists\vec{y}\psi(\vec{x},\vec{y})$ is

- *ground* when $|\vec{y}| = 0$,
- *atomic* when ground and $\psi(\vec{x})$ is a single atom $P(\vec{x})$, for some predicate P,
- an *instance query* when atomic and $|\vec{x}| = 1$,
- *Boolean* (BCQ) when $|\vec{x}| = 0$.

Example 4.1.1. The following are conjunctive queries

$$q_1(x_1, x_2) = \texttt{writes}(x_1, x_2) \wedge \texttt{Paper}(x_2) \tag{4.1}$$
$$q_2(x_1, x_2) = \texttt{publishedBy}(x_1, x_2) \tag{4.2}$$
$$q_3(x) = \texttt{Researcher}(x) \tag{4.3}$$
$$q_4 = \exists y_1 y_2(\texttt{writes}(lisa, y_2) \wedge \texttt{Paper}(y_2) \wedge \texttt{presentedAt}(y_2, y_1)) \tag{4.4}$$
$$q_5(x) = \exists y(\texttt{writes}(lisa, y) \wedge \texttt{Paper}(y) \wedge \texttt{presentedAt}(y, x)) \tag{4.5}$$

where (4.1) is a ground query retrieving any individual writing a paper, along with the paper; (4.2) is an atomic query computing all pairs of objects satisfying the predicate `publishedBy`; (4.3) is an instance query for `Researcher`; (4.4) is a Boolean query asking whether *lisa* presented any paper at any conference; and finally (4.5) is a generic CQ retrieving all conferences *lisa* presented a paper at.

Let $\mathcal{K}$ be a KB and $q(\vec{x}) = \exists\vec{y}\psi(\vec{x},\vec{y})$ a CQ. A *possible answer* for q w.r.t. $\mathcal{K}$ is a substitution σ mapping answer variables $\vec{x}$ to constants in $\mathcal{K}$. We sometimes represent the substitution σ as a vector of constants $\vec{a}$ such that $|\vec{a}| = |\vec{x}| = n$ and $\sigma = \{x_1 \mapsto a_1, \ldots, x_n \mapsto a_n\}$. A possible answer σ is an *answer under certain answer semantics* (*certain answer* for short) to q w.r.t. $\mathcal{K}$ if $\mathcal{K} \models (\exists\vec{y}\psi(\vec{x},\vec{y}))\sigma$. A possible answer σ is an *answer under ground semantics* (*ground answer* for short) to q w.r.t. $\mathcal{K}$ if there exists a substitution σ' from bound variables $\vec{y}$ to constants in $\mathcal{K}$ such that $\mathcal{K} \models \psi(\vec{x},\vec{y})\sigma\sigma'$. We denote the set of certain (resp. ground) answers as $\texttt{cert}(q,\mathcal{K})$ (resp. $\texttt{ground}(q,\mathcal{K})$). If q is a BCQ, then the set of certain answers is either an empty set or the set containing the identity substitution. An (un)satisfiability check can be seen as the special Boolean query $\bot$. In particular a KB is satisfiable iff $\texttt{cert}(\bot,\mathcal{K})$ is empty.

Conjunctive queries can be represented as a set of Datalog rules R_q of the form

$$R_q = \begin{cases} \emptyset & \text{if } q = \bot \\ \{\psi(\vec{x},\vec{y}) \rightarrow P_q(\vec{x})\} & \text{otherwise} \end{cases} \tag{4.6}$$

where P_q is a fresh predicate uniquely bound to q. Note that this allows us to define query answers by means of entailment of a single fact, i.e., $\vec{a} \in \mathtt{cert}(q, \mathcal{K})$ iff $\mathcal{K} \cup R_q \models P_q(\vec{a})$.

It is easy to see that certain answer semantics is equivalent to ground semantics when $\mathcal{K}$ is a Datalog KB, or q is a ground CQ. However, in general, this is not the case, and interpreting CQs under ground or certain answer semantics can lead to different results.

Example 4.1.2. Consider a KB consisting of a single axiom

$$\mathtt{Researcher} \sqsubseteq \exists\, \mathtt{writes}.\mathtt{Paper}$$

and $\{\mathtt{Researcher}(lisa), \mathtt{writes}(bart, thesis1)\}$ as ABox. Then, considering $q(x) = \exists y \mathtt{writes}(x, y)$, the set of answers under ground semantics is $\{bart\}$ while the set of answers under certain answer semantics is $\{lisa, bart\}$.

When considering CQs over OWL 2 KBs, we restrict ourselves to unary or binary predicates, representing, respectively, concept and role names.

Answering CQs w.r.t. KBs is computationally very hard and decidability for OWL 2 under certain answer semantics is still an open problem. There exists a number of classes of conjunctive queries for which CQ answering is known to be decidable over OWL 2.

Let q be a CQ. The *graph representation* $G_q = \langle V, E \rangle$ of q is an undirected (multi)graph where

- V is the set of terms in q, and
- E is the set of labelled edges $P(u, v)$, where u, v are terms in q, P is a binary predicate s.t. $P(u, v) \in q$.

A query q is *forest-shaped* if the subgraph obtained from G_q by removing all edges $P(u, v)$, s.t. u, v are both answer variables and P a predicate in q, is a forest rooted in answer variables of q. A *tree-shaped* query is a forest-shaped query with a single answer variable. Tree-shaped queries can be *rolled-up* [50] into single concepts.

Definition 4.1.1. *Let q be a tree-shaped query, and let x be the only free variable in q. Let $\mathtt{parent}(\cdot)$ be the function, induced by the tree underlying G_q, rooted in x, returning the* parent *of a given node. The rolled-up concept C_u for some term u in G_q is*

$$C_u = \prod_{C(u)\in q} C \sqcap \prod_{\substack{u=\mathtt{parent}(v)\\ R(u,v)\in q\\ v \text{ constant}}} \exists R.(\{v\} \sqcap C_v) \sqcap \prod_{\substack{u=\mathtt{parent}(v)\\ R(u,v)\in q\\ v \text{ variable}}} \exists R.C_v \tag{4.7}$$

The rolled-up concept of q is $C_q = C_x$.

Given $\mathcal{L}$ a DL language, a tree-shaped query q, and a fresh concept name D, q is said to be in $\mathcal{L}$ if $C_q \sqsubseteq D$ is a GCI in $\mathcal{L}$; moreover, answering q over a KB $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$ in $\mathcal{L}$ can be reduced to instance checking, i.e., $a \in \mathtt{cert}(q, \mathcal{K})$ iff $\langle \mathcal{O} \cup \{C_q \sqsubseteq D\}, \mathcal{A} \rangle \models D(a)$.

Internalisable queries are an extension of tree-shaped and forest-shaped queries, for which answering over OWL 2 KBs can be reduced to entailment of concept assertions and satisfiability [50] (hence the decidability of the problem).

Definition 4.1.2. *A CQ q is* internalisable *if its graph G_q does not contain cycles of length at least two involving only bound variables in q.*

Note that ground and forest-shaped queries are internalisable queries.

4.2 RDF and SPARQL

The *Resource Description Framework (RDF)* [98] is a framework for representing information on the Web. RDF allows expressing relations between resources by means of *triples*, i.e., statements of the form

$$\langle \mathtt{subject}, \mathtt{predicate}, \mathtt{object} \rangle$$

where `subject` and `object` represent the two resources being related and `predicate` is the resource representing the nature of their relationships.

A *resource* in RDF terms is either an *Internationalized Resource Identifier (IRI)* [23], a *literal* (numbers, strings, dates, enumerations, etc.) or a *blank node* (resources without a specific identifier, represented as `_:resource`).

Given an IRI `<http://example.com/resource>`, it can be abbreviated as `ex:resource`, with `ex` a unique identifier for the prefix `http://example.com/`. The prefix `ex` determines a *namespace* shared by all resources under the same prefix. Commonly used namespaces are `rdf:`, `rdfs:`, `owl:`, providing resources that define the specifications for RDF [98], the RDF schema (RDFS) [6] and OWL [77], respectively.

A collection of triples is usually stored in an RDF *store* and can be represented as a *graph* by interpreting subject and object of triples as nodes and predicates as edges.

An ABox can be represented as a set of triples, where a property assertion $A(b, c)$ corresponds to the triple $\langle b, A, c \rangle$, and a class assertion $A(b)$ corresponds to the triple $\langle b, \mathtt{rdf{:}type}, A \rangle$.

The SPARQL Protocol and RDF Query Language (SPARQL) [45] is a standard query language to retrieve and manipulate data stored as RDF triples. A SPARQL conjunctive query is an expression of the form

```
SELECT ?X1 ··· ?Xn
WHERE {
    S1 P1 O1.
    ···
    Sm Pm Om.
}
```

where variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$ (corresponding to the answer variables of a CQ) appear in the triple patterns ($\mathtt{S}_i\,\mathtt{P}_i\,\mathtt{O}_i$) composing the body of the query and where $\mathtt{S}_i$, $\mathtt{P}_i$, and $\mathtt{O}_i$ can be either RDF resources, blank nodes, or variables (represented by a leading question mark or a dollar sign). When representing BCQs (i.e., CQs without answer variables) the keywords `SELECT <variables> WHERE` are replaced by `ASK`.

Example 4.2.1. Conjunctive queries introduced in Example 4.1.1 can be expressed using SPARQL as follows. We represent existing concept and role names as resources under the namespace "`ex:`".

Query (4.1)

```
SELECT ?X ?Y
WHERE {
    ?X ex:writes ?Y.
    ?Y rdf:type ex:Paper
}
```

Query (4.2)

```
SELECT ?X ?Y
WHERE {
    ?X ex:publishedBy ?Y
}
```

Query (4.3)

```
SELECT ?X
WHERE {
    ?X rdf:type ex:Researcher
}
```

Query (4.4)

```
ASK {
    ex:lisa ex:writes ?Y.
    ?Y rdf:type ex:Paper
    ?Y ex:presentedAt ?X.
}
```

Query (4.5)

```
SELECT ?X
WHERE {
    ex:lisa ex:writes ?Y.
    ?Y rdf:type ex:Paper
    ?Y ex:presentedAt ?X.
}
```

The semantics of SPARQL queries over an RDF graph is based on subgraph matching [45]. Formally, a solution to a SPARQL CQ q is a mapping σ from variables and blank nodes in q to nodes in the queried RDF graph G, such that the application of σ to the body of the query results in a subgraph of G. An answer to a SPARQL CQ is the projection of σ over the answer variables.

Ontology axioms and data assertions can also be mapped into sets of RDF triples using an extended vocabulary defined by W3C standards, such as RDFS [6] and OWL [77]. The semantics of SPARQL queries over such extensions is obtained by redefining subgraph matching to take into account semantic entailment relations. Such extensions of the SPARQL semantics are called *entailment regimes* [37] and standardize the entailment relation in use, well-formed queries and graphs for the regime and the definition of entailment. The semantics of SPARQL queries w.r.t. OWL 2 ontologies is specified by the OWL 2 Direct Semantics entailment regime [37].

4.3 Computational Complexity

The decision problem associated with CQ answering is known as *conjunctive query entailment (CQE)*; given a KB $\mathcal{K}$, a query q and a possible answer σ, CQE consists in determining whether $\mathcal{K} \models q(\vec{x})\sigma$. As mentioned above, CQE is a well-known open problem, when considering unrestricted OWL DL and OWL 2 ontologies.

The problem is decidable for OWL DL ontologies when the query does *not* involve transitive relations [95]. Decidability of CQE is co–N2ExpTime-hard for $\mathcal{ALCHOIQ}$ [35] and 2–ExpTime–complete for DLs like $\mathcal{SHIQ}$ [36], $\mathcal{SHOQ}$ [32, 25], and for the Horn fragment of OWL 2, Horn-$\mathcal{SROIQ}$ [82]. Hardness results for 2–ExpTime have been shown even for weaker languages, such as $\mathcal{ALCI}$ [66] and $\mathcal{SH}$ [25]. We can lower the complexity to ExpTime–complete by removing inverse roles, e.g., in $\mathcal{ALC}$ and $\mathcal{SHQ}$ [66] or by considering the Horn fragment of OWL, Horn-$\mathcal{SHOIQ}$ [82]. When considering data complexity, CQE is coNP–complete for non-Horn DLs like $\mathcal{ALC}$ [97, 81], $\mathcal{ALE}$ [97] and $\mathcal{SHIQ}$ [36, 81] and P–complete for Horn-$\mathcal{SHOIQ}$ [83] and Horn-$\mathcal{SROIQ}$ [82]. CQE is ExpTime–complete for Horn-$\mathcal{ALCHOIQ}$ and NP–complete for RSA, while it is P–complete in data complexity for both languages [13, 29].

To further lower the complexity of the decision problem and find tractable ways of computing certain answers to CQs, less expressive fragments of OWL 2 (called OWL 2 *profiles* [72]) have been defined. Decidability of CQE for OWL 2 EL is PSpace–complete in combined complexity and P–complete in data complexity [62, 105]. The complexity drops to NP when excluding complex role inclusion (but maintaining reflexivity and transitivity) from the language [106]. Decidability of CQE in OWL 2 QL is NP–complete in combined complexity and AC^0 in data complexity [10, 12]. Decidability of CQE for OWL 2 RL is NP–complete in combined complexity and P–complete in data complexity [72]. The OWL 2 RL profile is a fragment of plain Datalog, for which CQE is P–complete in data complexity and ExpTime–complete in combined complexity [1]. When considering disjunctive Datalog the problem is coNP–complete in data complexity and coNExpTime–complete in combined complexity [18].

An alternative approach to obtain decidability of CQE is to consider CQs under ground semantics. *Ground conjunctive query entailment (GCQE)* is the problem of checking whether a substitution σ is a ground answer to a CQ $q(\vec{x})$ w.r.t. $\mathcal{K}$. GCQE can be easily reduced to satisfiability checking and hence, is decidable in OWL 2.

For more information on the computational complexity of decidability of CQE in other fragments of OWL 2, we refer the reader to [84].

4.4 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.

4.4.1 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 [100], HermiT [31], RacerPro [43] 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 [100] 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 [31] is a fully-fledged reasoner for OWL 2, based on the *hypertableau calculus* [78]; a SPARQL interface around HermiT is provided by OWL-BGP.[1]

RacerPro is a tableau–based system for the $\mathcal{SHIQ}$ DL language; it implements a technique for instance retrieval, called *filter and refine* [117] 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 [111], has been recently adapted to perform CQ answering over expressive ontologies [109], using an absorption-based technique [110, 108]. 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 [109], 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 [59] and data summarization [22].

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.

4.4.2 Materialization-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.

[1] `https://github.com/iliannakollia/owl-bgp`

Triple stores like Jena [70], Sesame [7] and Virtuoso [28] offer query answering capabilities over RDBMS and support the RDFS description language. OWLim [5] provides support for OWL 2 RL ontologies. A materialization-based reasoner extensively used in this work is RDFox [79], 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 [76, 73, 75, 79]. RDFox covers most of 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 [65], 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, materialization-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 [120] to effectively compute a sound lower bound to the set of certain answers to a CQ.

4.4.3 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).

Ontology-Based Data Access (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 [118] 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*[2] to transform the rewritten query into a query over the underlying relational data sources. The process is called *perfect reformulation* [89] 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.

[2] Often expressed in the W3C standard R2RML language [19].

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 FO 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 FO 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 [51, 58].

Rosati [94] applied the query rewriting technique to $\mathcal{EL}$ and $\mathcal{ELH}$, showing that unions of conjunctive queries (UCQs) can be rewritten into a Datalog query. The same result does not hold for $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$. REQUIEM [87, 88] 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 [27, 26] 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 [80]).

A different approach involves the manipulation and rewriting of the input query [32, 36]. 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 [50], the problem is reduced to testing the consistency of an extended KB.

4.4.4 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 KB. 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 [68, 60] in order to derive the certain answers to the input query. The technique has been applied to different description logics in the $\mathcal{EL}$ family, such as the extension of $\mathcal{ELH}$ with $\perp$ and range axioms [68] and $\mathcal{ELHO}^r_\perp$ [107], as well as in the DL-Lite family, e.g., DL-Lite$_{horn}$ with number restriction [60] and DL-Lite$_\mathcal{R}$ [67]. More recently the combined approach has been applied to RSA [14, 29] and its underlying ontology language Horn-$\mathcal{ALCHOIQ}$ [13].

In this thesis, we exploit the filtration-based combined approach for RSA [29] to compute bounds to the answers to an input query. In the following, we provide a brief overview of this technique.

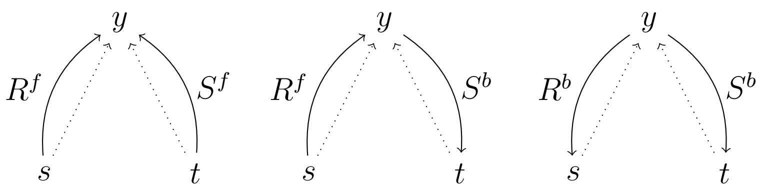
Combined approach for RSA

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 KB $\mathcal{K}$ is performed by computing the LHM of the definite program obtained by translating $\mathcal{K}$ according to the rules in Table 4.1. The translation is an enhanced version of the translation given in Table 3.2 where axioms of type (T5) are *Skolemized* if the role involved is unsafe, and *constant Skolemized* otherwise. Constant Skolemization of axioms can introduce *forks*, i.e., confluent chains of binary atoms, in the canonical model, possibly leading to spurious answers. Furthermore, the presence of inverse roles might create forks that lead to a spurious match even when the input query is linearly-shaped (see Fig. 4.1). In order to keep track of these forks, directionality is taken into account when Skolemizing an axiom; roles are annotated



$R(s,y) \wedge S(t,y)$ $\quad$ $R(s,y) \wedge S(y,t)$ $\quad$ $R(y,s) \wedge S(y,t)$

Figure 4.1: Forks in the presence of inverse roles. From left to right: forward/forward, forward/backward, backward/backward combinations of binary atoms.

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

Table 4.1: Translation of Horn-$\mathcal{ALCHOIQ}$ axioms to build $E_\mathcal{K}$.

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 matches 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 [29]. This is formalized in the definition of $E_\mathcal{K}$, the Datalog program used to compute the canonical model for $\mathcal{K}$.

Definition 4.4.1. *Let $\mathtt{confl}(R)$ 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 $\mathtt{self}(A,R,B)$ be the smallest set containing $v^{A,0}_{R,B}$ and $v^{A,1}_{R,B}$ if $R \in \mathtt{confl}(R)$; and let $\mathtt{cycle}(A,R,B)$ be the smallest set of terms containing, for each $S \in \mathtt{confl}(R)$,*

- $v^{D,0}_{S,C}$ *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* $\to \texttt{NI}(a)$ *for each constant* a *and all rules in Table 4.1. 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 axioms of type (T5) involving unsafe roles do not need to be constant Skolemized.

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

Theorem 4.4.1 ([29],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 a spurious answer induced 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 4.2. Filtering program $\mathcal{P}_q$ and its extension $\mathcal{P}_{q,\mathcal{K}}$ with $E_{\mathcal{K}}$ from Def. 4.4.1 are defined as follows.

Definition 4.4.2. *Let* $q = \exists \vec{y}.\psi(\vec{x}, \vec{y})$ *be a CQ, let* $\texttt{QM}$*,* $\texttt{sp}$*, and* $\texttt{fk}$ *be fresh predicates of arity* $|\vec{x}|+|\vec{y}|$*, let* $\texttt{id}$*,* $\texttt{AQ}^*$*,* $\texttt{TQ}^*$ *with* $* \in \{f, b\}$ *be fresh predicates of arity* $|\vec{x}|+|\vec{y}|+2$*, let* $\texttt{Ans}$ *be a fresh predicate of arity* $|\vec{x}|$*, let* $\texttt{named}$ *be a fresh unary predicate, and let* $\texttt{U}$ *be a set of fresh variables s.t.* $|U| \geq |\vec{y}|$*. Then,* $\mathcal{P}_q$ *is the smallest program with all rules in Table 4.2, and* $\mathcal{P}_{q,\mathcal{K}}$ *is defined as* $E_{\mathcal{K}} \cup \mathcal{P}_q$*.*

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

Table 4.2: Rules in $\mathcal{P}_Q$. Variables u, v, w from U are distinct.

Theorem 4.4.2 ([29], 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:*

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

(ii) $M[\mathcal{P}_{\mathcal{O},q}]$ is polynomial in $\lvert\mathcal{O}\rvert$ and exponential in $\lvert q\rvert$;

(iii) if $\mathcal{O}$ is satisfiable, $\vec{x} \in cert(q,\mathcal{O})$ iff $\mathcal{P}_{\mathcal{O},q} \models \mathtt{Ans}(\vec{x})$.

We can then build a worst-case exponential algorithm that, given an ontology $\mathcal{K}$ and a CQ q, 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 [29]. The algorithm first materializes $E_{\mathcal{K}}$ in polynomial time and then guesses a match σ for q over the materialization; finally it computes $(\mathcal{P}_{q,\mathcal{K}})\sigma$.

Theorem 4.4.3 ([29], 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.*

4.4.5 Hybrid approaches

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

Hydrowl [112] 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 [113] (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 [120] 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 [108] (inspired by the absorption technique presented by Steigmiller, Glimm, and Liebig [110]) 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 and complete for CQE for expressive ontology languages, such as $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$.

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. ACQuA uses a similar approach but tries to reduce the gap between upper and lower bounds by approximating to a more expressive language (RSA).

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 [50] under certain answer semantics [120] 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. Zhou [119] provides a more in-depth description of the algorithm and heuristics in use.

Given a KB $\mathcal{K} = \langle \mathcal{T} \cup \mathcal{R}, \mathcal{A} \rangle$[3] 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 for 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.

   (b) Using a variant of *shifting* [120, 24], $\mathcal{K}^{DD}$ 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 $\mathcal{ELHO}^r_\bot$ [105] 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 $\mathcal{ELHO}^r_\bot$ [68, 107] 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*.

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

(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* [21].

5. Once all spurious answers have been removed from G^q, $L^q \cup G^q$ is returned.

Let us 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 3.4.1). 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).[4] 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[5] 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 [45].

PAGOdA ensures that the returned answers are always *complete* under ground semantics, while being ultimately limited by the capabilities of HermiT when

[4] 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).

[5] `https://github.com/KRR-Oxford/PAGOdA`

considering the returned answers under certain answer semantics. HermiT does not natively support CQ answering and the process is first reduced to fact entailment. 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 *internalised* 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).

4.4.6 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 [99] and Val [115] in the context of logic theories (both propositional and FO logic) 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 [46], TrOWL [114] and SHER [22].

The SCREECH system [46] 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 [114] is a system providing CQ answering capabilities over OWL 2 DL. It uses a semantic approximation [86] technique to transform an OWL 2 DL ontology into OWL 2 QL for CQ answering and a syntactic approximation [91] from OWL 2 DL to OWL 2 EL for TBox reasoning. While being sound and complete for CQ answering, approximation 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 [22] 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 [21, 22]. 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, Möller, and Wessel [117]. 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 [34, 33] 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.

Finally, the recent work by Haga, Lutz, Sabellek, et al. [44] explores different notions of approximation for ontology-mediated queries over a selection of expressive languages like $\mathcal{ALC}$ and $\mathcal{ALCI}$. The authors aim at designing polynomial time approximations towards tractable languages like $\mathcal{ELI}$ or some restricted tuple–generating dependencies (TGDs) “from below and from above” (lower and upper bounds) with respect to CQs (and other query formalisms as well).

Part II

The ACQuA system

5

The hybrid approach of ACQuA

Contents

In this chapter we will provide the theoretical details behind our contributions. We first give an overview of the overall approach, and later go into details on the key components of this hybrid framework for CQ answering.

5.1 Overview

We propose a hybrid query answering architecture that combines black-box 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.

In particular, 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 compute 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. Any of these components could be potentially substituted or augmented with more capable ones; in particular, any relevant service mentioned in the previous sections 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 $\mathcal{K}$:

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.[1] Otherwise, proceed to step 3.

3. If $\mathcal{K}$ is RSA, return the full set of answers computed by RSAComb.[2] Otherwise, proceed to step 4.

4. Compute the bounds for 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.

[1]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.

[2]In this case, RSAComb provides a sound and complete algorithm for CQ answering over $\mathcal{K}$.

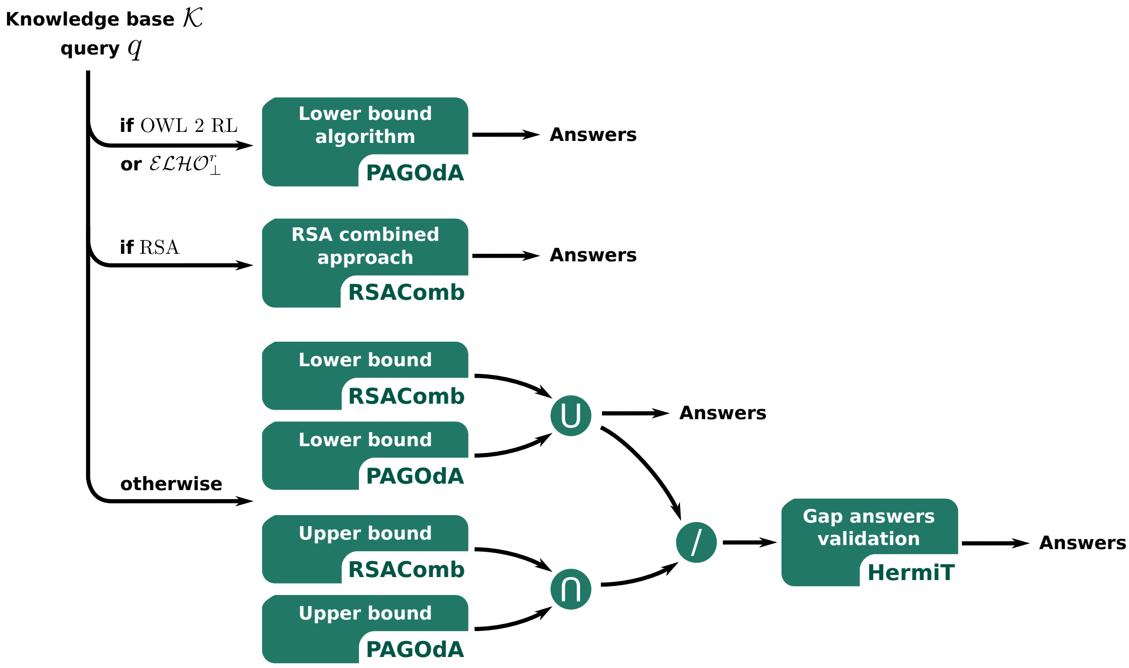


Figure 5.1: Workflow of the ACQuA system.

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 $L^q \cup G^q$.

A visual representation of these steps is given in Figure 5.1.

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 [29]. With the integration of PAGOdA, and a suitable fully-fledged reasoner, like HermiT, ACQuA is able to answer internalisable queries [50] over OWL 2 DL under certain answer semantics [120].

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 [120] for more details. We will instead focus our attention on the underlying RSAComb reasoner; in particular we dedicate Sections 5.2–5.3 to the description of the novel algorithms used in step 4 to compute L^q_R, U^q_R via approximation to RSA. In Chapter 6 we provide more details on the design

(a1)	$\texttt{PhDStudent}(bart)$	(a6)	$\texttt{writes}(bart, work1)$
(a2)	$\texttt{Researcher}(lisa)$	(a7)	$\texttt{writes}(lisa, work1)$
(a3)	$\texttt{Journal}(journal1)$	(a8)	$\texttt{published}(journal1, work1)$
(a4)	$\texttt{Journal}(journal2)$	(a9)	$\texttt{JournalPaper}(work1)$
(a5)	$\texttt{Journal}(journal3)$	(a10)	$\texttt{Report}(work1)$

(r1)	$\texttt{published} \equiv \texttt{publishedBy}^-$
(r2)	$\texttt{accepted} \sqsubseteq \texttt{reviewed}$
(t1)	$\texttt{PhDStudent} \sqsubseteq \texttt{Student} \sqcap \texttt{Researcher}$
(t2)	$\texttt{JournalPaper} \sqcap \texttt{Thesis} \sqsubseteq \bot$
(t3)	$\texttt{Report} \sqsubseteq \texttt{Paper} \sqcup \texttt{Thesis}$
(t4)	$\texttt{Journal} \sqsubseteq \exists\, \texttt{published}.\texttt{Paper}$
(t5)	$\texttt{Researcher} \sqsubseteq \exists\, \texttt{writes}.\texttt{Paper}$
(t6)	$\texttt{Paper} \sqsubseteq \exists\, \texttt{presentedAt}.\texttt{Conference}$
(t7)	$\texttt{Paper} \sqsubseteq \leq 1\, \texttt{presentedAt}.\texttt{Conference}$
(t8)	$\texttt{Conference} \sqsubseteq \exists\, \texttt{accepted}.\texttt{Paper}$
(t9)	$\exists\, \texttt{reviewed}^-.\texttt{Conference} \sqsubseteq \texttt{ConferencePaper}$

Table 5.1: Running example $\mathcal{K}_{ex}$.

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 5.1.1. Consider the KB $\mathcal{K}_{ex} = \langle \mathcal{T}_{ex} \cup \mathcal{R}_{ex}, \mathcal{A}_{ex} \rangle$, with ABox $\mathcal{A}_{ex}$ containing assertions (a1)–(a10), TBox $\mathcal{T}_{ex}$ containing axioms (t1)–(t9) and RBox $\mathcal{R}_{ex}$ containing axioms (r1)–(r2) in Table 5.1.

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 publication processes. Some axioms are not expressed in normal form (see Table 3.2) and can be further normalized as follows: axiom (t1) can be rewritten as

$$\texttt{PhDStudent} \sqsubseteq \texttt{Student} \tag{t1a}$$

$$\texttt{PhDStudent} \sqsubseteq \texttt{Researcher} \tag{t1b}$$

while axiom (r1) becomes

$$\texttt{published} \sqsubseteq \texttt{publishedBy}^- \tag{r1a}$$

$$\texttt{publishedBy} \sqsubseteq \texttt{published}^- \tag{r1b}$$

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

5.2 Lower bound computation

As mentioned in Section 4.4.5, PAGOdA computes 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.

In this section we present an alternative 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 substeps, 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* [120, 24];
3. From Horn-$\mathcal{ALCHOIQ}$ to RSA by modifying the input KB in order to enforce the constraints imposed by RSA (see Definition 3.4.3).

We are now going to explain these steps in more details.

5.2.1 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 3.2.

Let $\mathcal{K}'$ be the $\mathcal{ALCHOIQ}$ restriction of a KB $\mathcal{K}$. By the monotonicity of FO logic all certain answers w.r.t. $\mathcal{K}'$ are also certain answers w.r.t. $\mathcal{K}$. Moreover, if $\mathcal{K}'$ is unsatisfiable, so is $\mathcal{K}$.

In Example 5.1.1, $\mathcal{K}_{ex}$ is in $\mathcal{ALCHOIQ}$, so no axioms are discarded.

5.2.2 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.[3] 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* [24] 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 5.2.1. In Example 5.1.1, 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 $\mathcal{K}_{ex}$:

$$\texttt{JournalPaper} \sqsubseteq \overline{\texttt{Thesis}} \tag{5.1}$$

$$\texttt{Report} \sqcap \overline{\texttt{Thesis}} \sqsubseteq \texttt{Paper} \tag{5.2}$$

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

Program shifting is formally defined as follows.

Definition 5.2.1 ([120], 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 $\textit{\textbf{shift}}(r)$, is the following set of rules:*

- *if r is of the form*

$$\beta_1 \wedge \cdots \wedge \beta_n \to \bot \tag{5.3}$$

then

$$\textit{\textbf{shift}}(r) = \{r\} \cup \{\beta_1 \wedge \cdots \wedge \beta_{i-1} \wedge \beta_{i+1} \wedge \cdots \wedge \beta_n \to \bar{\beta}_i \mid 1 \leq i \leq n\} \tag{5.4}$$

- *if r is of the form*

$$\beta_1 \wedge \cdots \wedge \beta_n \to \gamma_1 \vee \cdots \vee \gamma_m \tag{5.5}$$

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

$$\beta_1 \wedge \cdots \wedge \beta_n \wedge \bar{\gamma}_1 \wedge \cdots \wedge \bar{\gamma}_m \to \bot \tag{5.6}$$

$$\beta_1 \wedge \cdots \wedge \beta_i \wedge \bar{\gamma}_1 \wedge \cdots \wedge \bar{\gamma}_{j-1} \wedge \bar{\gamma}_{j+1} \wedge \ldots \bar{\gamma}_m \to \gamma_j \quad \textit{for } 1 \leq j \leq m \tag{5.7}$$

$$\beta_1 \wedge \cdots \wedge \beta_{i-1} \wedge \beta_{i+1} \wedge \cdots \wedge \beta_n \wedge \bar{\gamma}_1 \wedge \ldots \bar{\gamma}_m \to \bar{\beta}_i \quad \textit{for } 1 \leq i \leq n \tag{5.8}$$

[3] 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:

$$\boldsymbol{shift}(\Sigma) = \bigcup_{r \in \Sigma} \boldsymbol{shift}(r) \tag{5.9}$$

We apply this technique to our $\mathcal{ALCHOIQ}$ KB in order to reduce it 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 5.2.1. *Let $\mathcal{K}' = \langle \mathcal{O}', \mathcal{A} \rangle$ be the $\mathcal{ALCHOIQ}$ restriction of the KB $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$, and let $\mathcal{K}'' = \langle \boldsymbol{shift}(\mathcal{O}'), \mathcal{A} \rangle$. Then $\boldsymbol{cert}(q, \mathcal{K}'') \subseteq \boldsymbol{cert}(q, \mathcal{K}')$.*

Proof (sketch). 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 $\texttt{shift}(\cdot)$, intuitively representing the complement of P. The following claims can be proved by induction on the derivation level of atoms in $\mathcal{M}$:

(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 $\texttt{shift}$ 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;

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 $\sigma \in \texttt{cert}(q, \mathcal{K}')$.

See Appendix A for a full version of the proof. $\square$

5.2.3 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 CQ q. Given a Horn-$\mathcal{ALCHOIQ}$ KB $\mathcal{K}$, checking if $\mathcal{K}$ is RSA consists of the following steps (see Def. 3.4.3):

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 FO logic, deleting axioms from $\mathcal{K}$ produces a lower bound approximation of $\mathcal{K}$ w.r.t. CQ answering.

Lemma 5.2.1. *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. 3.4.3 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 graphs, the 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 `E` to build $G_{\mathcal{K}}$. Finally, we use a modified depth-first search (DFS) visit (see Algorithm 1) of the graph to detect any cycle in $G_{\mathcal{K}}$; during the visit, the algorithm determines a representative node for each detected cycle, 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 $G_{\mathcal{K}}$

```
let N be the set of nodes in G_K;
let C be an empty set;
foreach node n in N do
    if n is not discovered then
        let S be an empty stack;
        push n to S;
        while S is not empty do
            pop v from S;
            if v is not discovered then
                label v as discovered;
                let adj be the set of nodes adjacent to v;
                if any node in adj is discovered then
                    push v to C;
                else
                    foreach node w in adj do
                        push w to S;
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 5.2.2. *Let $\mathcal{K}$ be a $\mathcal{SROIQ}$ KB, and $\mathcal{K}'$ its syntactic restriction to $\mathcal{ALCHOIQ}$. Then $cert(q, \mathtt{lower}(\mathtt{shift}(\mathcal{K}'))) \subseteq cert(q, \mathcal{K})$.*

Proof. By Section 5.2.1 and Theorem 5.2.1 we know that

$$\texttt{cert}(q, \texttt{shift}(\mathcal{K}')) \subseteq \texttt{cert}(q, \mathcal{K}') \subseteq \texttt{cert}(q, \mathcal{K}) \tag{5.10}$$

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

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*; this can potentially lead to a situation where PAGOdA is able to capture, e.g., some transitive knowledge, that is, on the other hand, ignored by RSAComb.

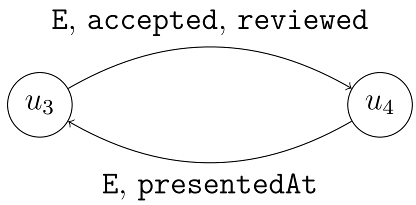


Figure 5.2: Graphical representation of $G_{\mathcal{K}_{ex}}$.

Example 5.2.2. Consider our running Example 5.1.1 and $\mathcal{K}'_{ex} = \mathtt{shift}(\mathcal{K}_{ex})$. Then

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

whereas all other roles are safe.

Now, let u_i, for $1 \leq i \leq 4$ be unique, fresh constants, and $\mathcal{P}^{ex}_{\mathrm{RSA}}$ be the logic program (according to Def. 3.4.3), corresponding to $\mathcal{K}'_{ex}$. In particular

$$\mathtt{Journal}(x) \rightarrow \mathtt{published}(x, u_1) \wedge \mathtt{PE}(x, u_1) \wedge \mathtt{Paper}(u_1) \tag{5.11}$$

$$\mathtt{Researcher}(x) \rightarrow \mathtt{writes}(x, u_2) \wedge \mathtt{PE}(x, u_2) \wedge \mathtt{Paper}(u_2) \tag{5.12}$$

$$\mathtt{Paper}(x) \rightarrow \mathtt{presentedAt}(x, u_3) \wedge \mathtt{PE}(x, u_3) \wedge \mathtt{Conference}(u_3) \wedge \mathtt{U}(u_3) \tag{5.13}$$

$$\mathtt{Conference}(x) \rightarrow \mathtt{accepted}(x, u_4) \wedge \mathtt{PE}(x, u_4) \wedge \mathtt{Paper}(u_4) \wedge \mathtt{U}(u_4) \tag{5.14}$$

is the translation (according to Def. 3.4.3) of (t4), (t5), (t6), and (t8). Finally, M^{ex}_{RSA} is the LHM of $\mathcal{P}^{ex}_{\mathrm{RSA}}$.

The dependency graph $G_{\mathcal{K}'_{ex}}$ is shown in Figure 5.2. $G_{\mathcal{K}'_{ex}}$ needs to be reduced to an oriented forest by detecting a set of nodes for removal. Let us 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 $\mathcal{K}''_{ex} = \mathtt{lower}(\mathcal{K}'_{ex})$.

Now, consider the query $q_l(x_2) = \mathtt{publishedBy}(x_1, x_2)$. Then,

$$\mathtt{cert}(q_l, \mathcal{K}''_{ex}) = \{journal1, journal2, journal3\}. \tag{5.15}$$

It can be verified that the lower bound computed by PAGOdA is not as tight and results in the set of answers $\{journal1\}$.

5.3 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. $cert(q, \mathcal{K}) \subseteq cert(q, \mathcal{K}')$ 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.

5.3.1 $\bot$ 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 FO logic, weakening the KB, it can be shown (see [120, 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.

5.3.2 Approximation of disjunctive rules

According to Table 3.2, 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 analyze 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 $\bot_f$; (one of) the furthest from $\bot_f$ is chosen. For more details on the definition of a choice function, see [120, Section 8.2].

Given a choice function `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 5.3.1. *Let δ be a function from axioms to axioms eliminating disjunction from the head of any axiom of type* (T1) *and* (T4)*:*

$$\delta(\alpha) = \begin{cases} \bigsqcap_{i=1}^{n} A_i \sqsubseteq \mathtt{ch}(\{B_j \mid 1 \leq j \leq m\}) & \textit{if } \alpha \equiv \bigsqcap_{i=1}^{n} A_i \sqsubseteq \bigsqcup_{j=1}^{m} B_j \\ A \sqsubseteq \leq 1R.B & \textit{if } \alpha \equiv A \sqsubseteq \leq mR.B \\ \alpha & \textit{otherwise} \end{cases} \tag{5.16}$$

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

Example 5.3.1. Consider axioms (t3) from our running example. Let `ch` be a choice function, such that

$$\mathtt{ch}(\{\mathtt{Paper}, \mathtt{Thesis}\}) = \mathtt{Paper} \tag{5.17}$$

Then $\delta(t3)$ is

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

5.3.3 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. 3.4.3, 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 $\alpha \equiv C \sqsubseteq \leq 1S.D$ 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 role S, $\{C \sqcap \exists S.D \sqsubseteq \bot\} \models C \sqsubseteq \leq 1S.D$ 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.[4] 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 5.2.3. 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 5.3.2. *We define δ' as a function from axioms to sets of axioms.*

$$\delta'(\alpha) = \begin{cases} \{C \sqcap \exists S.D \sqsubseteq \bot_f\} & \textit{if } \alpha \equiv C \sqsubseteq \leq 1S.D \textit{ and} \\ & \exists R \textit{ unsafe s.t. } R \sqsubseteq Inv(S) \textit{ and} \\ & w \approx t, R(t, u^A_{R,B}) \in M_{RSA} \\ & \textit{with } w, t \textit{ distinct} \\ \{A \sqsubseteq \exists R.\{b^A_{R,B}\}, \{b^A_{R,B}\} \sqsubseteq B\} & \textit{if } \alpha \equiv A \sqsubseteq \exists R.B \textit{ and } \texttt{marked}(\alpha) \\ \{\alpha\} & \textit{otherwise} \end{cases} \tag{5.18}$$

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

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

Theorem 5.3.1. *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 (sketch). The claims can be proven as follows:

(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. 5.3.2, is enough to render the knowledge base RSA$^+$.

(ii) To prove that $cert(q, \mathcal{K}) \subseteq cert(q, \mathcal{K}')$, it can be shown that, for every model $\mathcal{I}$ such that $\mathcal{I}$ is a model of the rewritten KB $\mathcal{K}'$, $\mathcal{I}$ is a model of $\mathcal{K}$.

(iii) Finally, assume $\vec{x} \in cert(q, \mathcal{K})$. By step (ii) we know that $\vec{x} \in cert(q, \mathcal{K}')$. Then, by definition of certain answer, there exists a match σ for q over $M[\pi(K)^{\approx,\top}]$. The claim can be proven by building a corresponding match σ' from σ over $M[\mathcal{P}_{\mathcal{K},q}]$, such that σ' is *non-anonymous*, *fork-free* and *acyclic*. It is, then, trivial to show that $\mathcal{P}_{\mathcal{K}',q} \models \texttt{Ans}(\vec{x})\sigma'$.

See Appendix A for a full version of the proof. □

Example 5.3.2. Consider, again, our running example (Example 5.1.1) and $\mathcal{K}'_{ex} = \delta(\mathcal{K}_{ex})$. Let $\mathcal{P}^{ex}_{\text{RSA}}$ be its translation into logic rules (according to Def. 3.4.3) and M^{ex}_{RSA} its LHM, as in Example 5.2.2. We know that the dependency graph $G_{\mathcal{K}_{ex}}$ (shown in Figure 5.2) is not an oriented forest. Let us 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. 4.4.1 (`accepted` is unsafe). We denote the KB resulting from this process with $\mathcal{K}''_{ex} = \texttt{upper}(\mathcal{K}'_{ex})$.

Now, consider the query

$$q_u(x_1, x_2) = \texttt{published}(x_1, x_3) \land \texttt{published}(x_2, x_3) \land x_1 \neq x_2 \tag{5.19}$$

Then,

$$\texttt{cert}(q_u, \mathcal{K}''_{ex}) = \emptyset. \tag{5.20}$$

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

$$\{\langle journal2, journal3\rangle, \langle journal3, journal2\rangle\} \tag{5.21}$$

5.3.4 Property chain axioms

Our tests show that, general property chain axioms (axioms of type (R4) in Table 3.2) are quite uncommon in practice.[5] 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 RSA [14, Theorem 2, Proposition 1]. Intuitively, due to monotonicity of FO logic, 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 [52]. 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.

[5] Over the 797 ontologies in the Oxford ontology repository (`http://krr-nas.cs.ox.ac.uk/ontologies/UID/`), only 82 ontologies ($\sim$10%) contain (up to 53) *complex* property chain axioms.

6
Design and architecture

Contents

We proposed a new framework to compute CQ answering over unrestricted OWL 2 DL ontologies by using answer bounds and further refinement steps. The approach has been implemented in a system called ACQuA [53], 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 and 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 [48] 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 4.4.4) 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 (see Sections 6.2–6.3). 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 5.1, along with a visual representation in Figure 5.1.

In this section we will describe some of the 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 [56] 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) 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 5.2–5.3.

The system is written in Scala and uses the OWLAPI [48] 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.

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[1], which provides a way to uniquely associate a sequence of terms with a fresh term;
- support for *named graphs* to isolate and cache partial computations;
- support for "TBox reasoning" in order to query the structure of an ontology represented as RDF triples.

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 system.

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 Chapter 7.

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.

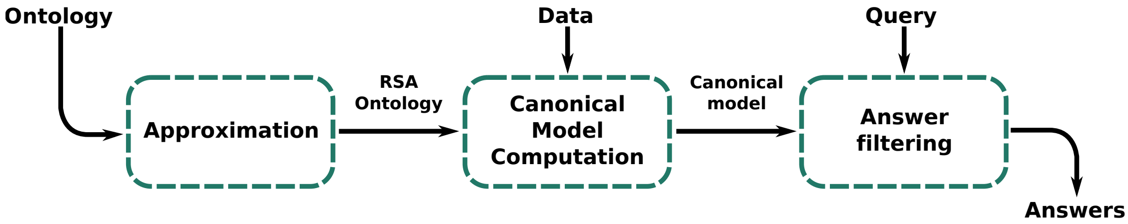


Figure 6.1: Workflow of the RSAComb system.

6.1.1 Overview

Figure 6.1 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 KB.

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 5.2; 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 KB. A copy of the ontology, translated into Datalog according to Def. 3.4.3, is imported into RDFox along with the data and materialized by the reasoner. The dependency graph and equality safety checks (see Definition 3.4.3) 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 5.3; 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. 4.4.1

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

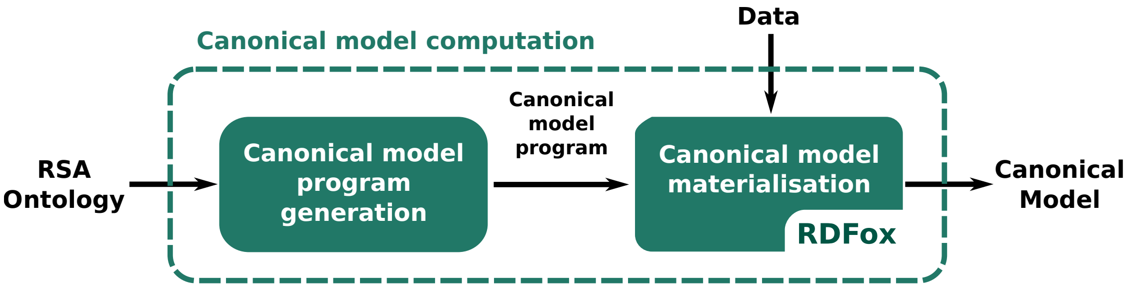


Figure 6.2: RSAComb: canonical model computation.

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, can be 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) and (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 involves the conversion of the input RSA ontology into logic rules as described in Def. 4.4.1, and where function symbols are simulated using RDFox's built-in Skolemization feature.

Example 6.1.1. A Skolemized rule derived from an existential axiom (T5)

$$A(x) \to R(x, f^A_{R,B}(x)) \land B(f^A_{R,B}(x)) \tag{6.1}$$

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;[2] in the case of equality axiomatization, the feature cannot be enabled along other features like *aggregates* and *negation-as-failure* (with the latter used in the filtering step).

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[?Z,?Z] .
```

The `notIn` and `named` predicates

Our work also includes a few clarifications on theoretical definitions and their implementation. In the canonical model computation [29], 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

[2]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.

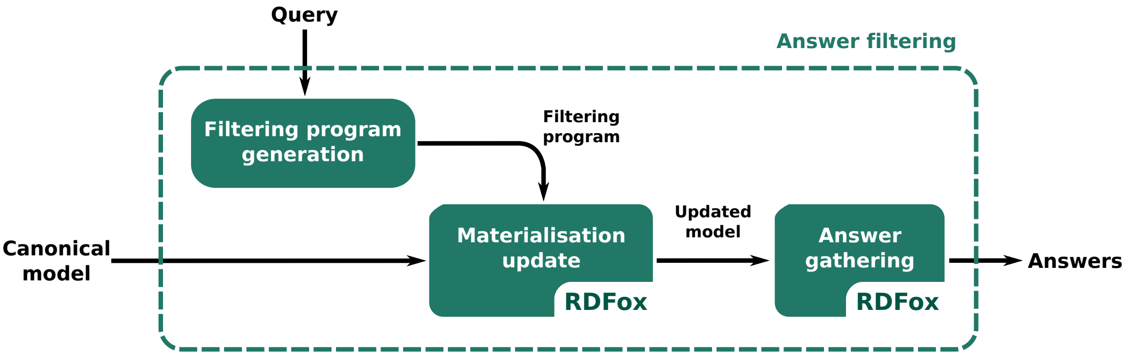


Figure 6.3: RSAComb: answer filtering.

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. 6.3, 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 [29] 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*.

Example 6.1.2. 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 4.2) 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.

$$id(\vec{x}, \vec{y}, u, v), id(\vec{x}, \vec{y}, v, w) \rightarrow id(\vec{x}, \vec{y}, u, w) \tag{6.2}$$

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 $1 \leq n \leq arity(P)$, are fresh predicates of arity 2. Rule (3c) can be *reified* as:

$$\begin{aligned} id_1(k, x_1), \ldots, id_{m+n}(k, y_n), id_{m+n+1}(k, u), id_{m+n+2}(k, v),& \\ id_1(j, x_1), \ldots, id_{m+n}(j, y_n), id_{m+n+1}(j, v), id_{m+n+2}(j, w),& \\ l := \texttt{KEY}(\vec{x}, \vec{y}, u, w) \rightarrow id_1(l, x_1), \ldots, id_{m+n}(l, y_n),& \\ id_{m+n+1}(l, v), id_{m+n+2}(l, w)& \end{aligned} \tag{6.3}$$

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 (see predicate *id* in (6.4)) without having to introduce $arity(P)$ fresh predicates. 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 $\texttt{SKOLEM}(u, v, j)$ 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 $\langle u, v \rangle$, respectively. Joins over multiple terms (*id* joining over $(\vec{x}, \vec{y})$ in (6.2)) can now be rewritten into simpler joins (*id* joining over a single term k).[3]

$$id(k, j), \texttt{SKOLEM}(u, v, j), id(k, l), \texttt{SKOLEM}(v, w, l), \texttt{SKOLEM}(u, w, t) \rightarrow id(k, t) \tag{6.4}$$

□

[3]Rule 6.4 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.

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

Table 6.1: Improved rules for the filtering step for the RSA combined approach.

The complete rewriting of the filtering program is provided in Table 6.1. According to the documentation[4] 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 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

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

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 6.1.3. 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

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

6.2 Lower bound approximation to RSA

As described in Section 5.2, 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 (Def. 5.2.1);
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 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.

We first translate the axioms in the knowledge base according to Definition 3.4.3, 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.[5]

RDFox supports importing OWL 2 axioms and the conversion into RDF triples is performed automatically. RBox reasoning (Listing 6.1) is then achieved by importing the following rules into the RDF store.

```
1  [?X,rdfs:subPropertyOf,?Y],
2  [?Y,rdfs:subPropertyOf,?X] :-
3     [?X,owl:equivalentProperty,?Y].
4
5  [?Yi,rdfs:subPropertyOf,?Xi] :-
6      [?X,rdfs:subPropertyOf,?Y],
7      [?Xi,owl:inverseOf,?X],
8      [?Yi,owl:inverseOf,?Y] .
9
10 [?Y,owl:inverseOf,?X] :-
11     [?X,owl:inverseOf,?Y] .
12
13 [?X,rdfs:subPropertyOf,?X],
14 [?Y,rdfs:subPropertyOf,?Y] :-
15     [?X,rdfs:subPropertyOf,?Y].
16
17 [?X,:subPropertyOfTrans,?Y] :-
18     [?X,rdfs:subPropertyOf,?Y].
19
20 [?X, :subPropertyOfTrans, ?Z] :-
21    [?X, :subPropertyOfTrans, ?Y],
22    [?Y, :subPropertyOfTrans, ?Z] .
```

Listing 6.1: Rules for role subsumption reasoning

These encode reflexivity and transitivity of sub-role axioms (R2) (lines 13–22), taking into account inverse (lines 5–11) and equivalent roles (lines 1–3), as well.

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

Once both the data and the axioms have been imported and materialized according to their respective rules, the equality safety condition (i) of Definition 3.4.3 can be formulated as a query as follows:

```
1  SELECT ?A ?S ?B WHERE {
2      ?W owl:sameAs ?T .
3      filter ( ?W != ?T ) .
4      ?T ?R [ a rsa:U ] .
5      ?R rdfs:subPropertyOf ?Si .
6      ?Si owl:inverseOf ?S .
7      ?X rdf:type owl:Restriction .
8      ?X owl:onProperty ?S .
9      ?X maxQualifiedCardinality "1" .
10     ?X owl:onClass ?B .
11     ?A rdfs:subClassOf ?X .
12 }
```

Listing 6.2: Condition 1 of equality safety in the 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 $A \sqsubseteq \leq 1S.B$ 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 as follows:

```
1  SELECT ?R ?P WHERE {
2    ?A ?R ?U .
3    ?U ?S ?A .
4    ?A a rsa:NI .
5    ?U a rsa:U .
6    ?R rdfs:subPropertyOf ?P .
7    FILTER ( ?R != ?P ) .
8    ?P rdfs:subPropertyOfTrans ?T .
9    ?T owl:inverseOf ?Ti .
10   ?S rdfs:subPropertyOfTrans ?Ti .
11 }
```

Listing 6.3: Condition 2 of equality safety in the RSA definition

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 subproperties from R to T, making the knowledge base equality safe.

6.3 Upper bound approximation to RSA

The approximation algorithm proposed in Section 5.3 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.

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 5.3.2).

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 knowledge base according to Def. 5.3.2, 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 5.3.1 the answers produced by RSAComb are an upper bound to the answers to the query.

7 Evaluation

Contents

We provide here an extensive evaluation over a range of benchmark ontologies. We start by looking at some performance results for RSAComb [56], our implementation of the combined approach for RSA, followed by a comparison of our system ACQuA [53] with PAGOdA.[1] This latter comparison is first carried out by providing an analysis of the execution times of the two tools over a selection of test cases. Additionally, we provide a more qualitative comparison over the number of gap answers that need further processing, and the related number of calls to an underlying fully-fledged reasoner. To draw this fine-grained analysis we chose to test ACQuA against PAGOdA, which shares a similar approach to CQ answering, while leaving out other approaches, such as the absorption-based query entailment implemented in Konclude [108, 109] (see Section 4.4).

Section 7.1 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 available online [54].

[1] `https://github.com/KRR-Oxford/PAGOdA` (commit `8651164c`)

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.

7.1 Benchmarks

We use two different sets of benchmark ontologies:

- the *PAGOdA batch* mimics the evaluation process originally performed for the RSA combined approach [29] and for PAGOdA [120];
- the *Oxford ontology repository (OOR) batch* is a subset of the Oxford ontology repository[2], and it is 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.[3] 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, a realistic ontology for which both data and relevant queries are provided. To test scalability, the datasets of this ontology have been sampled in subsets of increasing size.

A summary of the statistics regarding each of these ontologies can be found in Table 7.1 where n is the parameter passed to the data generator for LUBM and UOBM.

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.[4] 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 roles and existential patterns from the structure of the ontology. In order to generate a suitable number of queries we used the following step:

[2] `http://krr-nas.cs.ox.ac.uk/ontologies/UID/`
[3] `https://www.cs.ox.ac.uk/isg/tools/PAGOdA/2015/jair/`
[4] `http://krr-nas.cs.ox.ac.uk/ontologies/readme.htm`

	# 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

Table 7.1: Benchmarks statistics, with LUBM/UOBM data generators depending on a parameter n.

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 14 135 concept atomic queries, 4 434 role atomic queries and 3 893 existential queries for a total of 22 462 queries over 126 ontologies. Apart from the basic atomic patterns, we chose to include existential queries of the form $q(x) = \exists y[R(x, y) \wedge B(y)]$, for some role R and concept B, because of the potentially different results given when considering the query under ground and certain answer semantics and the fact that, in preliminary testing, we noticed that PAGOdA was having some difficulties returning a sound set of answers to queries of this shape.

Example 7.1.1. LUBM TBox contains the following axiom describing the fact that each research assistant works for at least one research group

$$\texttt{ResearchAssistant} \sqsubseteq \exists \texttt{worksFor.ResearchGroup} \tag{7.1}$$

The following query

```
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).[5]

[5]PAGOdA guarantees a sound and complete set of answers under certain answers semantics if the bounds match or the query can be *internalized* into a DL concept. Otherwise, it 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).

The collection of queries and the scripts to generate them are part of our benchmark distribution [54].

7.2 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 particularly usefully since we were able to draw a very close comparison between the two tools and improve upon the observations provided by Zhou, Cuenca Grau, Nenov, et al. [120].

7.2.1 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 [57]. Given that the original reference implementation [29] 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 the ones provided by the original paper.

Our implementation is written in Scala and uses RDFox[6] 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 [48] 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.

In the following we provide test results of our system against LUBM [42] and Reactome[7] using the set of queries originally used by Feier, Carral, Stefanoni, et al. [29] and available in Appendix C. All results provided below are averages of at least 3 measurements.

In Figure 7.1 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.

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

[6] `https://www.oxfordsemantic.tech/product`

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

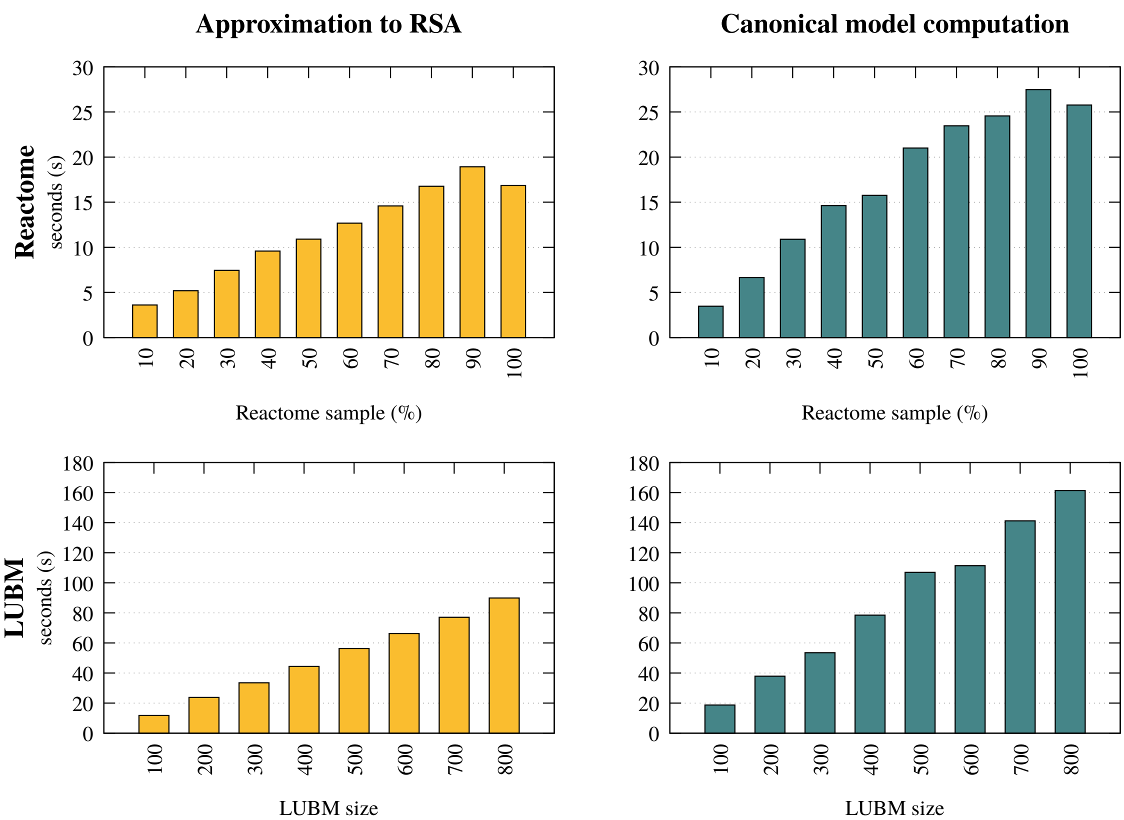


Figure 7.1: Scalability of approximation to RSA and canonical model computation in RSAComb.

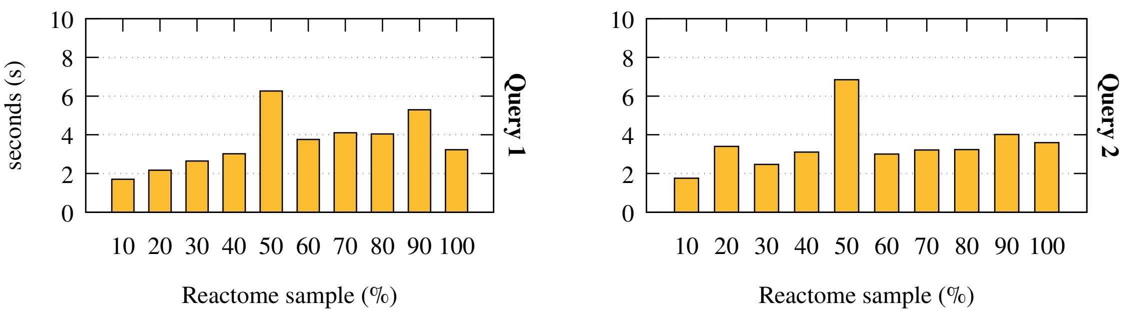


Figure 7.2: RSAComb answer filtering in Reactome.

not necessarily correspond to a greater amount of filtering, as shown in Figure 7.2, 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 Figure 7.4).

This unpredictability of the filtering step can "backfire" when a huge amount of filtering is involved. In Figure 7.3 we show the filtering time for query 2 in

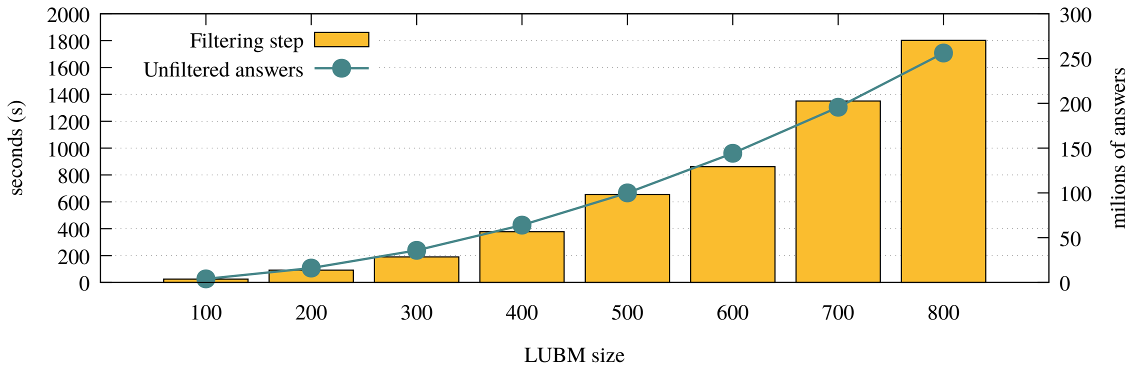


Figure 7.3: Answer filtering with high degree of filtration in Query 2 in LUBM.

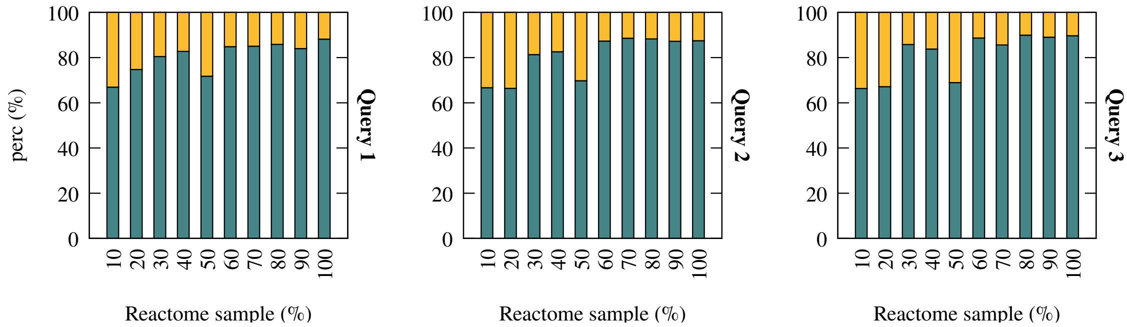


Figure 7.4: Percent time distribution of canonical model computation (at the bottom, in blue) and answer filtering (at the top, in yellow) in Reactome.

LUBM along with the amount of unfiltered answers that the filtering program needs to process. Of these, less than 1‰ is found to be part of the certain answers. Figure 7.3 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 huge filtering step, processing hundreds of millions of facts.

Finally, Figure 7.4 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.

7.2.2 ACQuA

We will now provide test results for the PAGOdA batch, a subset of the benchmarks initially used to evaluate the performance of PAGOdA [120]. 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.

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 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.

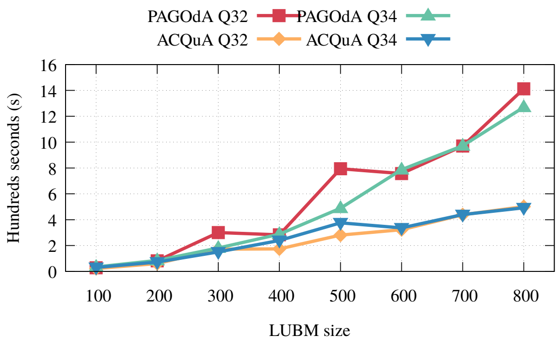
In general, ACQuA is able to match PAGOdA's results, as well as performance, in all queries in the (G1-2) groups. This should not come as a surprise, since the results from PAGOdA were not leaving much room for improvement and were showing that more complex CQ answering techniques were not needed for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

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

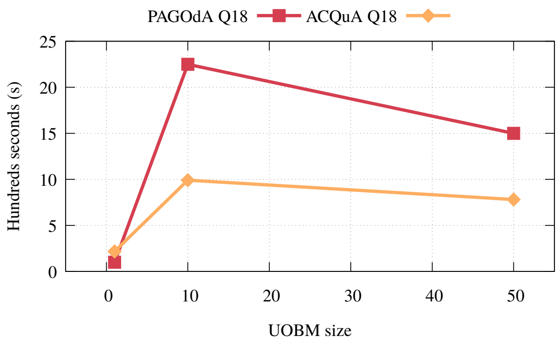
According to [120, Section 10.3.2], PAGOdA falls back to HermiT in the following queries to compute the correct set of answers: queries 32 and 34 in LUBM, query 18 in UOBM (for some data sizes) and query 65 in Reactome. Figure 7.5 sums up the results for our tests. Pre-processing times for the ontology are not taken into account here since the process is common to both tools and, in general, can be computed offline.

LUBM

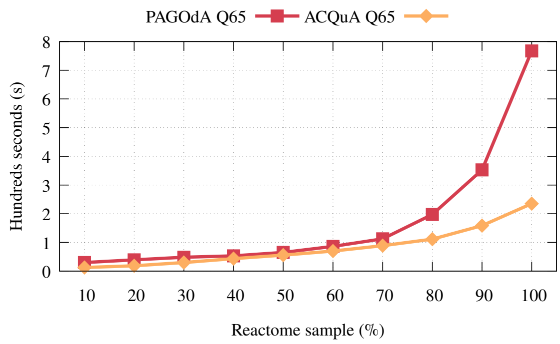
For both query 32 and 34, PAGOdA computes an *empty* lower bound and an exact upper bound, leaving a gap between the bounds of 26 and 14 possible answers, respectively, with just as many calls to HermiT. ACQuA, is able to compute a matching lower bound, avoiding the calls to HermiT altogether. This resulted in a significant improvement on the query processing time (Figure 7.5a). 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.



(a) LUBM query processing.



(b) UOBM query processing.



(c) Reactome query processing.

Figure 7.5: Scalability of query processing times for LUBM, UOBM and Reactome in ACQuA vs PAGOdA.

UOBM

We could not perform a direct comparison with UOBM since we were unable to fully reproduce the results shown by Zhou, Cuenca Grau, Nenov, et al. [120], with PAGOdA not terminating within the provided time limit of 10h for bigger sizes of data. Regardless, we were able to observe a recognizable pattern in the results for query 18. In Figure 7.5b, we report our results against an estimate of PAGOdA's performance determined by looking at the graphs by Zhou, Cuenca Grau, Nenov, et al. [120].

In this case, PAGOdA is able to compute a tight lower bound, while showing a gap in the answers in the order of thousands, caused by the upper bound computation. Even in this case we were able to avoid the use of HermiT by computing a matching upper bound, consequently improving the query processing time overall.

Reactome

In query 65 of Reactome, PAGOdA fails to compute matching bounds, presenting a gap of up to 52 answers that requires the use of a full reasoner. In our case, we were

	Ontologies processed	Queries executed	Non-empty queries
PAGOdA	103	18 235	1 455
ACQuA	126	22 462	2 256

Table 7.2: PAGOdA and ACQuA statistics on OOR batch (over 126 ontologies and 22 462 queries).

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.[8] In these cases ACQuA was able to fix the issue and compute the sound and complete set of answers under certain answer semantics.

7.3 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 7.2, PAGOdA was able to process 103 out of 126 ontologies considered, executing around 81% of the generated queries; out of these, only 1455 (circa 8%) have a non-empty answer set. ACQuA, on the other hand, was able to process the entire set of ontologies, answering the full suite of generated queries, of which around 10% have a non-empty answer set.

We identified a set of 18 (role atomic) queries over DOLCE [30] 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, avoiding the use of HermiT, overall. 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 $\mathcal{SHOIN}(\mathcal{D})$ (see Table 7.3).

In Table 7.4 we provide quantitative and performance results for the queries over ontology 24, where we denote the lower bound, upper bound and query processing

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

DOLCE	expressivity	# axioms	# facts
00014	$\mathcal{SHOIN}(D)$	1544	119
00024	$\mathcal{SHOIN}(D)$	1544	137

Table 7.3: Statistics for DOLCE fragments 14 and 24 from the OOR.

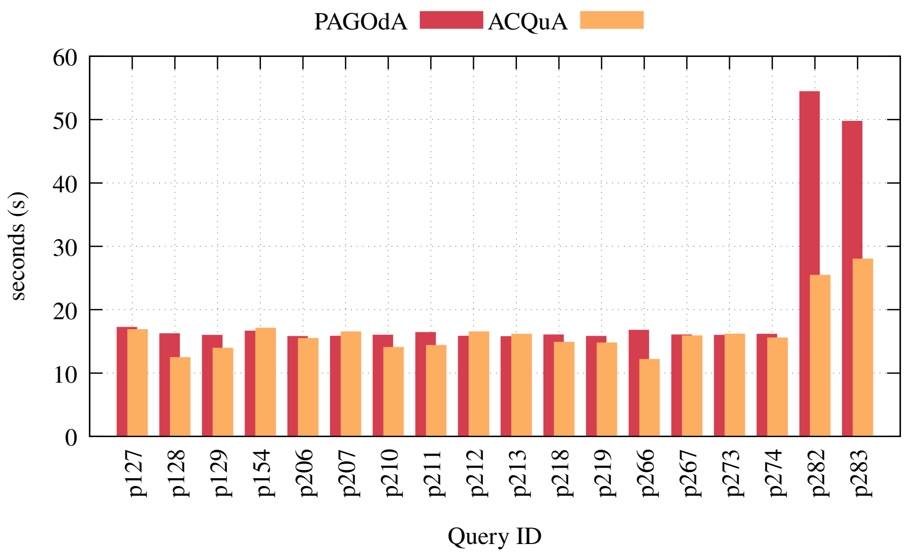


Figure 7.6: Execution time on DOLCE queries in PAGOdA (red) vs ACQuA (orange).

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. It can be seen from the table that PAGOdA computes a tight lower bound to the answers to the queries, and this is inherited in ACQuA by using the former tool as an intermediate step. On top of this, the use of RSAComb in ACQuA makes it possible to compute a matching upper bound.

In the first 16 queries we obtained comparable performance results (see Figure 7.6). 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, which reduces the total number of calls to HermiT to 0.

Finally, we found a set of 23 queries across multiple ontologies for which PAGOdA returned an unsound set of answers. We were able to fix the issue in ACQuA, and return the correct answers to the queries under certain answer semantics.

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

Query ID	PAGOdA L	PAGOdA U	PAGOdA T	ACQuA L	ACQuA U	ACQuA T	Total 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

Table 7.4: Results of ACQuA vs PAGOdA on DOLCE.

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;
- The filtering step in RSAComb is able to handle millions of triples;
- ACQuA is able to outperform PAGOdA in a selection of test cases, improving *both* the lower and upper bounds;
- ACQuA is able to fix some performance issues present in PAGOdA, by computing matching bounds and hence further limiting the use of HermiT;
- The ability to avoid the use of HermiT when computing matching bounds results in significant performance improvements.

8
Discussion and conclusions

In this work, we presented a new hybrid query answering architecture that combines black-box 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. The technique is based on the computation of answer bounds "from above" and "from below" and their progressive refinement to compute the full set of certain answers. To this end, we propose two novel algorithms to compute lower and upper bounds to the answers to a query via approximation to RSA and RSA$^+$, respectively. These techniques led to the development of two new systems:

- RSAComb, an efficient implementation of the combined approach for RSA [29], reorganized to fit the new implementation design and the integration of RDFox as the underlying 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 of the hybrid architecture combining RSAComb, PAGOdA [120], and HermiT [31] 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, while enjoying 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 provided an extensive evaluation of the systems, first testing scalability and performance of RSAComb as a standalone system and then, comparing ACQuA against PAGOdA.

In ACQuA, 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. This might not always be true, and, in general, the proposed architecture involves a trade-off performance analysis concerning the addition of new reasoners to the picture.

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. This comparison reports on differences on the execution time, showing how ACQuA provides, in general, better performance than PAGOdA, and on qualitative results, analysing the number of gap answers and calls to HermiT in both tools. As we mentioned in Chapter 5, these qualitative improvements are the result of the combination of the (incomparable) bounds computed with PAGOdA and RSAComb, with the performance of the two tools varying extensively depending on the KB and query in input. During our evaluation process we identified different scenarios in which the contribution of the two tools towards the final result greatly differed, justifying the inclusion of both tools in ACQuA.

We intend to further extend this work in a few different directions. In this work we made extensive use of RSA, and introduced RSA$^+$, an extension of the language enriched with axioms to represent (ir)reflexivity, asymmetry and disjointness among roles. We proved that the combined approach for RSA is still complete when applied to this extension. We think that a more thorough analysis of the expressive power of the language would be beneficial for a further refinement of the approximation algorithms. In particular, it would be interesting to analyze the correctness of the combined approach for RSA applied to RSA$^+$ to show whether the additional axioms increase the expressivity of the language.

On top of this, alternative approximation techniques targetting RSA could be explored. The handling of transitive properties in the approximation "from below" might, for example, be refined by using a technique like *box-pushing*; such a rewriting does not preserve entailment of CQ, but might be enough to provide lower bound guarantees on the computation of query answers.

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, Kazakov, Kollia, et al. [33].

On a similar note, relationships between queries have not been considered so far and might be beneficial to the computation of the final set of answers. To this end, it might be possible to consider dependences between sets of queries and exploit execution order and cached partial results to compute answers more efficiently.

From a practical standpoint, the integration of different tools may also introduce programming languages and license compatibility issue, especially when building a commercial application. While different tools might provide their own integration interfaces, the proposed architecture might benefit from a unified interface (e.g., a REST API) that can be wrapped around external tools, regardless of their internal implementation details.

Finally, *singularization* [69] is an alternative to equality axiomatization that replaces the equality predicate $\approx$ with a fresh predicate $\mathtt{Eq}$. The new predicate is defined as an equivalence relation but lacks the substitution rules (see Eq. 2.4); the premises of rules in a KB are instead rewritten to compensate for the lack of substitution rules. Properties of singularization have been thoroughly explored in the literature [69, 39], and adapting the combined approach for RSA to work with this technique might lead to better performance in practice.

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.

Appendices

A
Proofs

This chapter provides proofs for lemmas and theorems used in Sections 5.2–5.3.[1]

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 $s \leq t$ ($s < t$) 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.

[1]Some of the following lemmas are adapted from drafts of proofs for the theorems in [29]. These drafts were provided by the authors but, to the best of our knowledge, have never been published.

Theorem 5.2.1. *Let $\mathcal{K}' = \langle \mathcal{O}', \mathcal{A} \rangle$ be the $\mathcal{ALCHOIQ}$ restriction of the KB $\mathcal{K} = \langle \mathcal{O}, \mathcal{A} \rangle$, and let $\mathcal{K}'' = \langle \texttt{shift}(\mathcal{O}'), \mathcal{A} \rangle$. Then $\texttt{cert}(q, \mathcal{K}'') \subseteq \texttt{cert}(q, \mathcal{K}')$.*

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 $\texttt{shift}(\cdot)$, intuitively representing the complement of P. In order to prove the theorem, we introduce the following claims:

(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 $\texttt{shift}$ 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 $1 \leq i \leq n$. If $r \in \mathcal{K}''$, then, by definition of $\texttt{shift}$, $r \in \mathcal{K}'$. Moreover, by IH, we have $\mathcal{K}' \models A_i(c)$ for $1 \leq i \leq n$, 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 $\texttt{shift}$ for some predicate P in the signature of $\mathcal{K}'$. Since $\overline{P}(c) \notin \mathcal{A}$, there must be a rule

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

with $A_i(c) \in \mathcal{M}$ for $1 \leq i \leq n$ and $\overline{B}_i(c) \in \mathcal{M}$ for $1 \leq i \leq m$. By IH, $\mathcal{K}' \models A_i(c)$ for $1 \leq i \leq n$ and $\mathcal{K}' \not\models B_i(c)$ for $1 \leq i \leq m$. 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 $1 \leq i \leq n$, $\overline{B}_i(c) \in \mathcal{M}$ for $1 \leq i \leq m$. By IH, $\mathcal{K}' \models A_i(c)$ for $1 \leq i \leq n$ and $\mathcal{K}' \not\models B_i(c)$ for $1 \leq i \leq m$. Moreover, if $r \in \mathcal{K}''$, there must be a rule $r' \in \mathcal{K}'$ of the form

$$\bigwedge_{i=1}^{n} A_i(x) \rightarrow \bigvee_{i=1}^{m} B_i(x) \vee P(x) \tag{A.2}$$

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*

$$\eta_c(t) = \begin{cases} t & \textit{if } t \in N_I \\ u^A_{R,B} & \textit{if } t \textit{ is of type } (A, R, B) \end{cases} \tag{A.3}$$

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 $\eta_u : terms(\mathcal{M}_u) \rightarrow terms(M_{RSA})$ be the following function*

$$\eta_u(t) = \begin{cases} t & \textit{if } t \in N_I \\ u^A_{R,B} & \textit{if } t \textit{ is of type } (A, R, B) \end{cases} \tag{A.4}$$

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) $Inv(R)(y,x) \rightarrow R(x,y)$ with $Inv(R)(t_2,t_1) \in \mathcal{M}_c$. As in the previous case, we have $level(Inv(R)(t_2,t_1),\mathcal{M}_c) < level(R(t_1,t_2),\mathcal{M}_c)$. It can be easily shown that $Inv(R)$ fulfills all hypotesis 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) $R(x,y) \wedge y \approx z \rightarrow R(x,z)$ 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) $R(x,y) \wedge x \approx z \rightarrow R(z,y)$ 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) $\top(x) \to x \approx x$ 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) $A(x) \to x \approx a$, 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) $A(x) \wedge S(x, y) \wedge B(y) \wedge S(x, z) \wedge B(z) \to y \approx z$. Moreover, there 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) $x \approx y \to y \approx x$ 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) $x \approx y \wedge y \approx z \to x \approx z$ 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.

$\square$

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 $\mathcal{M}_u$. $\square$

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

$$\bigwedge_{i=1}^{m} A_i(t_i) \wedge \bigwedge_{j=1}^{n} R_j(u_{1j}, u_{2j}) \tag{A.5}$$

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

$$\bigwedge_{i=1}^{m} A_i(t_i) \wedge \bigwedge_{j=1}^{n} R_j^{a_j}(u_{1j}, u_{2j}) \tag{A.6}$$

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

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

$$\bigwedge_{i=1}^{m} A_i(t_i) \wedge \bigwedge_{j=1}^{n} R_j^{a_j}(u_{1j}, u_{2j}) \tag{A.7}$$

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

$$\bigwedge_{i=1}^{m} A_i(t_i) \wedge \bigwedge_{j=1}^{n} L_j \tag{A.8}$$

where

$$L_j = \begin{cases} R(u_{1j}, u_{2j}) & \text{if } a_j = \sqcup \\ R^f(u_{1j}, u_{2j}) & \text{if } a_j = f \\ Inv(R)^f(u_{2j}, u_{1j}) & \text{if } a_j = b \end{cases} \tag{A.9}$$

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* $\mathtt{named}(\lambda(x)) \in \mathcal{M}$ *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* $R^f(u, y_i), S^f(v, y_j) \in (\psi(\vec{x}, \vec{y}))^{\vec{a}}_n$*, 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*

$$R^f_{o_1}(y_{l_1}, y_{l_2}), R^f_{o_2}(y_{l_3}, y_{l_4}), \ldots, R^f_{o_p}(y_{l_{2p-1}}, y_{l_{2p}}) \in (\psi(\vec{x}, \vec{y}))^{\vec{a}}_n \tag{A.10}$$

such that

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

Lemma A.5. *For a given substitution* $\lambda : \vec{x} \to terms(\mathcal{M})$*, it is the case that* $\mathcal{M} \models \mathtt{Ans}(\lambda(\vec{x}))$ *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 6.1), 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:*

$$\sigma(t) = \begin{cases} \xi(t) & \text{if } \xi(t) \in N_I \\ f(\sigma(u)) & \text{if } \xi(t) = f(u) \text{ for some function symbol } f \text{ in } \mathcal{M}_u \end{cases} \tag{A.11}$$

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

$$\theta(t) = \begin{cases} t & \text{if } t \in N_I \\ f^A_{R,B}(\theta(u)) & \text{if } t = f^A_{R,B}(u) \text{ and } R \text{ is unsafe} \\ v^{A,0}_{R,B} & \text{if } t = f^A_{R,B}(u),\ R \text{ is safe and } \theta(u) \notin \mathtt{unfold}(A,R,B) \\ v^{A,i+1}_{R,B} & \text{if } t = f^A_{R,B}(u),\ R \in \mathtt{confl}(R) \text{ and } \theta(u) = v^{A,i}_{R,B}, \text{ for } i = 0,1 \\ v^{A,1}_{R,B} & \text{if } t = f^A_{R,B}(u) \text{ and } \theta(u) \in \mathtt{cycle}(A,R,B) \end{cases} \tag{A.12}$$

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

$$depth_*(t) = \begin{cases} 0 & \text{if } \xi(t) \in N_I \\ 1 + depth_*(u) & \text{if } \xi(t) = f(u) \end{cases} \tag{A.13}$$

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).

$\square$

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*

$$R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c \tag{A.14}$$

(ii) $R(t_1, t_2) \in \mathcal{M}_u$, $\sigma(t_1) < \sigma(t_2)$ *and* $\sigma(t_1) \in N_I$ *implies*

$$R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c \text{ or } R^b(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c \tag{A.15}$$

(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*

$$R^b(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c \tag{A.16}$$

(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*

$$R^b(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c \tag{A.17}$$

(v) $R(t_1, t_2) \in \mathcal{M}_u$*,* $\sigma(t_2) < \sigma(t_1)$ *and* $\sigma(t_2) \in N_I$ *implies*

$$R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c \text{ or } R^b(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c \tag{A.18}$$

(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*

$$R^f(\theta(\sigma(t_1)), \theta(\sigma(t_2))) \in \mathcal{M}_c \tag{A.19}$$

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) $A(x) \to R(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B}(x))$ 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

$$A(x) \to R^f(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B}(x)) \tag{A.20}$$

    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

$$A(x) \wedge \mathtt{notIn}(x, \mathtt{unfold}(A, R, B)) \to R^f(x, v^{A,0}_{R,B}) \wedge B(v^{A,0}_{R,B}) \tag{A.21}$$

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

(b) $S(x,y) \to R(x,y)$ 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 $S^f(x,y) \to R^f(x,y)$, thus $R^f(\theta(\sigma(t_1)),\theta(\sigma(t_2))) \in \mathcal{M}_c$.

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

(d) $R(x,y), z \approx y \to R(x,z)$ 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) $R(x,y), z \approx x \to R(z,y)$ 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 $\pi(\mathcal{K})^{\approx,\top}$:

(a) $A(x) \to R(x, f^A_{R,B}(x)) \wedge B(f^A_{R,B}(x))$, 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) $S(x,y) \to R(x,y)$ — similar to case (i)b.

(c) $Inv(R)(y,x) \to R(x,y)$ — similar to case (i)c.

(d) $R(x,y) \wedge y \approx z \to R(x,z)$ — similar to case (i)d.

(e) $R(x,y) \wedge x \approx z \to R(z,y)$ — similar to case (i)e.

(iv) $a = R(t_1,t_2) \in \mathcal{M}_u$, with $\sigma(t_2) < \sigma(t_1)$, and $\sigma(t_2) \notin N_I$ — similar to case (iii).

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

(vi) $a = R(t_1,t_2) \in \mathcal{M}_u$, with $\sigma(t_2) \not< \sigma(t_1)$, $\sigma(t_2) \in N_I$ 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, $depth_c(t_1) = depth_c(t_2)$. □

$$\begin{tikzcd}
t_1 \arrow[d, dotted, bend right, "R^f"'] & \approx & s \arrow[d, "T^f"', "T^b"] & \approx & t_3 \arrow[d, dotted] \\
t_2 & \approx & t & \approx & t_4 \arrow[u, dotted, bend right, "S^f"']
\end{tikzcd}$$

Figure A.1: Ambiguous roles in $\mathcal{M}_c$ in which both $T^f(s,t)$ and $T_b(s,t)$ hold.

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 4.4.1. □

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. A.1).

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., 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

$$\begin{cases} t_1 = v^{D,0}_{S,C} \\ t_2 = v^{A,1}_{R,B} \end{cases} \quad or \quad \begin{cases} t_1 = v^{D,1}_{S,C} \\ t_2 = v^{A,0}_{R,B} \end{cases} \tag{A.22}$$

and

$$\begin{cases} t_3 = v_{S,C}^{D,0} \\ t_4 = v_{R,B}^{A,0} \end{cases} \quad or \quad \begin{cases} t_3 = v_{S,C}^{D,1} \\ t_4 = v_{R,B}^{A,1} \end{cases} \tag{A.23}$$

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 $\psi(\vec{x}, \vec{y})$:*

$$a_j = \begin{cases} \sqcup & \textit{if } R_j(\lambda(u_{1j}), \lambda(u_{2j})) \in \mathcal{M}_c \textit{ and} \\ & R_j^f(\lambda(u_{1j}), \lambda(u_{2j})), R_j^b(\lambda(u_{1j}), \lambda(u_{2j})) \notin \mathcal{M}_c \\ f & \textit{if } R_j^f(\lambda(u_{1j}), \lambda(u_{2j})) \in \mathcal{M}_c \\ b & \textit{if } R_j^b(\lambda(u_{1j}), \lambda(u_{2j})) \in \mathcal{M}_c \end{cases} \tag{A.24}$$

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

1. $id(\lambda(\vec{x}), \lambda(\vec{y}), l_{2i}, l_{2i+1}) \in \mathcal{M}$ for every $1 \leq i \leq p$ where $l_{2p+1} = l_1$;

2. $\texttt{NI}(\lambda(y_{l_j})) \notin \mathcal{M}$ for every $1 \leq j \leq 2p$.

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

$$R_{o_1}(s_p, s_1), R_{o_2}(s_1, s_2), \ldots, R_{o_p}(s_{p-1}, s_p) \in \mathcal{M}_u \tag{A.25}$$

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

To see that $(\lambda, \vec{a})$ is fork-free, we assume again the contrary. Then, there must be a pair of axioms $R^f(u, y_i), S^f(v, y_j) \in (\psi(\vec{x}, \vec{y}))^{\vec{a}}_n$, 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 $\mathtt{NI}(\lambda(y_i))$, $\mathtt{NI}(\lambda(y_j)) \notin \mathcal{M}$, it follows that $\mathtt{NI}(\sigma(\rho(y_i))), \mathtt{NI}(\sigma(\rho(y_j))) \notin \mathcal{M}_u$, $\sigma(\rho(y_i)) = f_{R,B}(\sigma(\rho(u)))$ 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 $1 \leq i \leq m$.

  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 $1 \leq i \leq m$ and that $\sigma(\rho(u)) = \sigma(\rho(v))$. Finally, we obtain $\lambda(u) = \lambda(v)$ — contradiction.

□

Theorem 5.3.1. *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. 5.3.2, 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 $\mathcal{K}$.[2]

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 $\alpha \equiv C \sqsubseteq \leq 1S.D \in \mathcal{K}$ 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 $\alpha \equiv A \sqsubseteq \leq mR.B \in \mathcal{K}$ has been rewritten into $\beta \equiv A \sqsubseteq \leq 1R.B$. 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 $B = \mathtt{ch}(\{B_j \mid 1 \leq j \leq m\})$. 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

$$\lambda'(t) = \begin{cases} \rho(t) & \text{for every } t \in terms(q) \setminus \vec{y} \\ t & \textit{otherwise} \end{cases} \tag{A.26}$$

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 $\mathcal{P}_{\mathcal{K}',q} \models \mathtt{Ans}(\lambda(\vec{x}))$.

□

[2] 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 $\mathcal{K}$.

B
Naming convention for DLs

A DL language definition determines the constructors and axioms available in its syntax. In order to distinguish between different DLs, a mnemonic naming convention has been introduced in the DL community; as such, a DL language is usually named according to a basic DL, and by adding letters and symbols according to the concept constructors, role constructors and axioms available. Table B.1 [4] introduces the symbols associated with the different language features mentioned in this work, and the definition of three basic DL languages, i.e., $\mathcal{AL}$, $\mathcal{EL}$, and $\mathcal{S}$.

Note that the definition of the symbol $^+$ is non-standard and it can appear in literature with the generic meaning of "additional features". In this work we will adopt the definition proposed by Motik, Shearer, and Horrocks [78].

[1] Includes disjoint (`Dis`), (ir)reflexive (`Ref`, `Irr`), (a)symmetric (`Sym`, `Asy`) roles.

Description	Syntax	Sym	$\mathcal{AL}$	$\mathcal{EL}$	$\mathcal{S}$
Top	$\top$		✓	✓	✓
Bottom	$\bot$	$_\bot$	✓		✓
Conjunction	$C \sqcap D$		✓	✓	✓
Atomic negation	$\neg A$		✓		✓
Value restriction	$\forall R.C$		✓		✓
Disjunction	$C \sqcup D$	$\mathcal{U}$			✓
Negation	$\neg C$	$\mathcal{C}, ^\neg$			✓
Existential restriction	$\exists R.C$	$\mathcal{E}$		✓	✓
Unqualified number restriction	$\leq nR$ $\geq nR$	$\mathcal{N}$			
Qualified number restriction	$\leq nR.C$ $\geq nR.C$	$\mathcal{Q}$			
Nominal	$\{a\}$	$\mathcal{O}$			
Range	$\texttt{range}(R, C)$	^{r}			
Inverse role	R^-	$\mathcal{I}$			
Role inclusion	$R \sqsubseteq S$	$\mathcal{H}$			
Complex role inclusion	$R_1 \circ \cdots \circ R_m \sqsubseteq S$	$\mathcal{R}$			
Functionality	$\texttt{Func}(R)$	$\mathcal{F}$			
Transitivity	$\texttt{Trans}(R)$	$_{R^+}$			✓
Additional roles[1]	(see note)	$^+$			
Datatype properties		(D)			

Table B.1: Naming convention for DLs, with $A \in N_C$, $a \in N_I$, $n, m \in \mathbb{N}$, C, D concepts, $R_1, \ldots, R_m, R, S$ roles.

C
Benchmark queries

Following are the queries initially introduced by Feier, Carral, Stefanoni, et al. [29] and used in this work as part of the benchmark for RSAComb.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT *
WHERE {
    ?X rdf:type :Student .
    ?X :takesCourse ?Y .
    ?Z rdf:type :Student .
    ?Z :takesCourse ?Y .
    ?Y rdf:type :Course .
    ?X :advisor ?Z .
    ?Z :advisor ?W .
}
```

Listing C.1: LUBM ontology, query 1.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT *
WHERE {
    ?X :headOf ?Y .
    ?Z :headOf ?Y .
    ?Y rdf:type :Department .
}
```

Listing C.2: LUBM ontology, query 2.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT *
WHERE {
    ?X rdf:type :Student .
    ?X :takesCourse ?Y .
```

```
    ?Y rdf:type :Course .
    ?Y :teacherOf ?Z .
}
```

Listing C.3: LUBM ontology, query 3.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT *
WHERE {
    ?X rdf:type :Professor .
    ?Y :publicationAuthor ?X .
    ?Y rdf:type :Publication .
    ?Y :memberOf ?Z .
    ?Z rdf:type :Department .
}
```

Listing C.4: LUBM ontology, query 4.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.biopax.org/release/biopax-level3.owl#>
SELECT ?X ?Y ?Z
WHERE {
    ?X rdf:type :Pathway .
    ?X :pathwayComponent ?Y .
    ?Y rdf:type :BiochemicalReaction .
    ?Y :participant ?Z .
    ?Z rdf:type :Protein .
}
```

Listing C.5: Reactome ontology, query 1.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.biopax.org/release/biopax-level3.owl#>
SELECT ?X ?Y ?Z
WHERE {
    ?X rdf:type :Pathway .
    ?X :pathwayComponent ?Y .
    ?Y rdf:type :BiochemicalReaction .
    ?Y :participant ?Z .
    ?Z rdf:type :Complex .
}
```

Listing C.6: Reactome ontology, query 2.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://www.biopax.org/release/biopax-level3.owl#>
SELECT ?X ?Y ?Z ?W
WHERE {
    ?X :participantStoichiometry ?Y .
    ?Y :physicalEntity ?Z .
    ?Z :participantStoichiometry ?W .
    ?W :physicalEntity ?Z .
}
```

Listing C.7: Reactome ontology, query 3.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. URL: `http://webdam.inria.fr/Alice/`.

[2] Andrea Acciarri, Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Mattia Palmieri, and Riccardo Rosati. "QuOnto: Querying Ontologies". In: *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*. Ed. by Manuela M. Veloso and Subbarao Kambhampati. AAAI Press / The MIT Press, 2005, pp. 1670–1671. URL: `http://www.aaai.org/Library/AAAI/2005/isd05-001.php`.

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[4] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[5] Barry Bishop, Atanas Kiryakov, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. "OWLIM: A family of scalable semantic repositories". In: *Semantic Web* 2.1 (2011), pp. 33–42. URL: `https://doi.org/10.3233/SW-2011-0026`.

[6] Dan Brickley and Ramanathan Guha. *RDF Schema 1.1*. W3C Recommendation. https://www.w3.org/TR/2014/REC-rdf-schema-20140225/. W3C, Feb. 2014.

[7] Jeen Broekstra, Arjohn Kampman, and Frank 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, Sardinia, Italy, June 9-12, 2002, Proceedings*. Ed. by Ian Horrocks and James A. Hendler. Vol. 2342. Lecture Notes in Computer Science. Springer, 2002, pp. 54–68. URL: `https://doi.org/10.1007/3-540-48005-6%5C_7`.

[8] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. "DL-Learner - A framework for inductive learning on the Semantic Web". In: *J. Web Semant.* 39 (2016), pp. 15–24. URL: `https://doi.org/10.1016/j.websem.2016.06.001`.

[9] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo 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] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. "Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family". In: *Journal of Automated Reasoning* 39.3 (2007), pp. 385–429.

[11] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. "The MASTRO system for ontology-based data access". In: *Semantic Web* 2.1 (2011), pp. 43–53. URL: https://doi.org/10.3233/SW-2011-0029.

[12] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. "Data complexity of query answering in description logics". In: *Artif. Intell.* 195 (2013), pp. 335–360. URL: https://doi.org/10.1016/j.artint.2012.10.003.

[13] David Carral, Irina Dragoste, and Markus 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*. Ed. by Michael Thielscher, Francesca Toni, and Frank Wolter. AAAI Press, 2018, pp. 339–348. URL: https://aaai.org/ocs/index.php/KR/KR18/paper/view/18076.

[14] David Carral, Cristina Feier, Bernardo Cuenca Grau, Pascal Hitzler, and Ian Horrocks. "Pushing the Boundaries of Tractable Ontology Reasoning". In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*. Vol. 8797. Lecture Notes in Computer Science. Springer, 2014, pp. 148–163.

[15] Werner Ceusters, Barry Smith, and L Goldberg. "A Terminological and Ontological Analysis of the NCI Thesaurus". In: *Methods of information in medicine* 44 (Feb. 2005), pp. 498–507.

[16] Pierre Chaussecourte, Birte Glimm, Ian Horrocks, Boris Motik, and Laurent Pierre. "The Energy Management Adviser at EDF". In: Oct. 2013, pp. 49–64.

[17] Sofia Cramerotti and Dario Ianes. "An Ontology-based System for Building Individualized Education Plans for Students with Special Educational Needs". In: *Procedia - Social and Behavioral Sciences* 217 (Feb. 2016), pp. 192–200.

[18] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. "Complexity and expressive power of logic programming". In: *ACM Comput. Surv.* 33.3 (2001), pp. 374–425. URL: https://doi.org/10.1145/502807.502810.

[19] Souripriya Das, Seema Sundara, and Richard Cyganiak. *R2RML: RDB to RDF Mapping Language*. W3C Recommendation. https://www.w3.org/TR/2012/REC-r2rml-20120927/. W3C, Sept. 2012.

[20] Sebastian Derriere, André Richard, and Andrea Preite-Martinez. "An ontology of sstronomical object types for the Virtual Observatory". In: *Proceedings of the International Astronomical Union* 2 (Aug. 2006), pp. 603–603.

[21] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Aaron Kershenbaum, Edith Schonberg, Kavitha Srinivas, and Li Ma. “Scalable Semantic Retrieval through Summarization and Refinement”. In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada.* AAAI Press, 2007, pp. 299–304. URL: `http://www.aaai.org/Library/AAAI/2007/aaai07-046.php`.

[22] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Edith Schonberg, and Kavitha Srinivas. “Scalable highly expressive reasoner (SHER)”. In: *J. Web Semant.* 7.4 (2009), pp. 357–361. URL: `https://doi.org/10.1016/j.websem.2009.05.002`.

[23] Martin Dürst and Michel Suignard. *Internationalized Resource Identifiers (IRIs).* RFC 3987. https://www.rfc-editor.org/rfc/rfc3987.html. Jan. 2005.

[24] Thomas Eiter, Michael Fink, Hans Tompits, and Stefan 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.* Ed. by Didier Dubois, Christopher A. Welty, and Mary-Anne Williams. AAAI Press, 2004, pp. 447–458. URL: `http://www.aaai.org/Library/KR/2004/kr04-047.php`.

[25] Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Simkus. “Query Answering in Description Logics with Transitive Roles”. In: *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009.* 2009, pp. 759–764.

[26] Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. “Query Rewriting for Horn-SHIQ Plus Rules”. In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.* Ed. by Jörg Hoffmann and Bart Selman. AAAI Press, 2012. URL: `http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4931`.

[27] Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui 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.* Ed. by Yevgeny Kazakov, Domenico Lembo, and Frank Wolter. Vol. 846. CEUR Workshop Proceedings. CEUR-WS.org, 2012. URL: `http://ceur-ws.org/Vol-846/paper%5C_20.pdf`.

[28] Orri Erling and Ivan Mikhailov. “Virtuoso: RDF Support in a Native RDBMS”. In: *Semantic Web Information Management - A Model-Based Perspective.* Ed. by Roberto De Virgilio, Fausto Giunchiglia, and Letizia Tanca. Springer, 2009, pp. 501–519. URL: `https://doi.org/10.1007/978-3-642-04329-1%5C_21`.

[29] Cristina Feier, David Carral, Giorgio Stefanoni, Bernardo Cuenca Grau, and Ian 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.

[30] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. "Sweetening Ontologies with DOLCE". In: *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings*. Ed. by Asunción Gómez-Pérez and V. Richard Benjamins. Vol. 2473. Lecture Notes in Computer Science. Springer, 2002, pp. 166–181. URL: https://doi.org/10.1007/3-540-45810-7%5C_18.

[31] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. "HermiT: An OWL 2 Reasoner". In: *Journal of Automated Reasoning* 53.3 (2014), pp. 245–269.

[32] Birte Glimm, Ian Horrocks, and Ulrike 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*. Ed. by Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris. Vol. 250. CEUR Workshop Proceedings. CEUR-WS.org, 2007. URL: http://ceur-ws.org/Vol-250/paper%5C_63.pdf.

[33] Birte Glimm, Yevgeny Kazakov, Ilianna Kollia, and Giorgos B. Stamou. "Lower and Upper Bounds for SPARQL Queries over OWL Ontologies". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 109–115. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9715.

[34] Birte Glimm, Yevgeny Kazakov, Ilianna Kollia, and Giorgos 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*. Ed. by C. Maria Keet and Valentina A. M. Tamma. Vol. 1265. CEUR Workshop Proceedings. CEUR-WS.org, 2014, pp. 1–12. URL: http://ceur-ws.org/Vol-1265/owled2014%5C_submission%5C_1.pdf.

[35] Birte Glimm, Yevgeny Kazakov, and Carsten Lutz. "Status QIO: An Update". In: *Proceedings of the 24th International Workshop on Description Logics (DL 2011), Barcelona, Spain, July 13-16, 2011*. Ed. by Riccardo Rosati, Sebastian Rudolph, and Michael Zakharyaschev. Vol. 745. CEUR Workshop Proceedings. CEUR-WS.org, 2011. URL: http://ceur-ws.org/Vol-745/paper%5C_44.pdf.

[36] Birte Glimm, Carsten Lutz, Ian Horrocks, and Ulrike Sattler. "Conjunctive Query Answering for the Description Logic SHIQ". In: *Journal of Artificial Intelligence Research* 31 (2008), pp. 157–204.

[37] Birte Glimm and Chimezie Ogbuji. *SPARQL 1.1 Entailment Regimes*. W3C Recommendation. https://www.w3.org/TR/2013/REC-sparql11-entailment-20130321/. W3C, Mar. 2013.

[38] Christine Golbreich. "The Foundational Model of Anatomy in OWL: Experience and Perspectives." In: Jan. 2005.

[39] Bernardo Cuenca Grau, Ian Horrocks, Markus Krötzsch, Clemens Kupke, Despoina Magka, Boris Motik, and Zhe Wang. "Acyclicity Notions for Existential Rules and Their Application to Query Answering in Ontologies". In: *J. Artif. Intell. Res.* 47 (2013), pp. 741–808. URL: https://doi.org/10.1613/jair.3949.

[40] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter F. Patel-Schneider, and Ulrike Sattler. "OWL 2: The next step for OWL". In: *J. Web Semant.* 6.4 (2008), pp. 309–322. URL: https://doi.org/10.1016/j.websem.2008.05.001.

[41] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan 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.

[42] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. "LUBM: A benchmark for OWL knowledge base systems". In: *Journal of Web Semantics* 3.2-3 (2005), pp. 158–182. URL: https://doi.org/10.1016/j.websem.2005.06.005.

[43] Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. "The RacerPro knowledge representation and reasoning system". In: *Semantic Web* 3.3 (2012), pp. 267–277. URL: https://doi.org/10.3233/SW-2011-0032.

[44] Anneke Haga, Carsten Lutz, Leif Sabellek, and Frank 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. URL: https://doi.org/10.24963/kr.2021/31.

[45] Steven Harris and Andy Seaborne. *SPARQL 1.1 Query Language.* W3C Recommendation. https://www.w3.org/TR/2013/REC-sparql11-query-20130321/. W3C, Mar. 2013.

[46] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and Tuvshintur Tserendorj. "Approximate OWL Instance Retrieval with SCREECH". In: *Logic and Probability for Scene Interpretation, 24.02. - 29.02.2008.* Ed. by Anthony G. Cohn, David C. Hogg, Ralf Möller, and Bernd Neumann. Vol. 08091. Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2008. URL: http://drops.dagstuhl.de/opus/volltexte/2008/1615/.

[47] Robert Hoehndorf, Michel Dumontier, and Georgios Gkoutos. "Evaluation of research in biomedical ontologies". In: *Briefings in bioinformatics* 14 (Sept. 2012).

[48] Matthew Horridge and Sean Bechhofer. "The OWL API: A Java API for OWL ontologies". In: *Semantic Web* 2.1 (2011), pp. 11–21. URL: https://doi.org/10.3233/SW-2011-0025.

[49] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. "The Even More Irresistible SROIQ". In: *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2-5, 2006.* Ed. by Patrick Doherty, John Mylopoulos, and Christopher A. Welty. AAAI Press, 2006, pp. 57–67. URL: http://www.aaai.org/Library/KR/2006/kr06-009.php.

[50] Ian Horrocks and Sergio Tessaris. "A Conjunctive Query Language for Description Logic Aboxes". In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA*. Ed. by Henry A. Kautz and Bruce W. Porter. AAAI Press / The MIT Press, 2000, pp. 399–404. URL: `http://www.aaai.org/Library/AAAI/2000/aaai00-061.php`.

[51] Dag Hovland, Roman Kontchakov, Martin G. Skjæveland, Arild Waaler, and Michael Zakharyaschev. "Ontology-Based Data Access to Slegge". In: *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*. Vol. 10588. Lecture Notes in Computer Science. Springer, 2017, pp. 120–129.

[52] Pan Hu, Boris Motik, and Ian 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. URL: `https://doi.org/10.1609/aaai.v33i01.33012859`.

[53] Federico Igne, Stefano Germano, and Ian Horrocks. *ACQuA - A hybrid framework providing a CQ answering service for OWL*. Version v0.2.1. May 2022. URL: `https://doi.org/10.5281/zenodo.6564388`.

[54] Federico Igne, Stefano Germano, and Ian Horrocks. *Benchmarks and scripts for ACQuA and RSAComb*. Zenodo, May 2022. URL: `https://doi.org/10.5281/zenodo.6564995`.

[55] Federico Igne, Stefano Germano, and Ian 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, Virtual Event, October 24-28, 2021, Proceedings*. Ed. by Andreas Hotho, Eva Blomqvist, Stefan Dietze, Achille Fokoue, Ying Ding, Payam M. Barnaghi, Armin Haller, Mauro Dragoni, and Harith Alani. Vol. 12922. Lecture Notes in Computer Science. Springer, 2021, pp. 200–216. URL: `https://doi.org/10.1007/978-3-030-88361-4%5C_12`.

[56] Federico Igne, Stefano Germano, and Ian Horrocks. *RSAComb - Combined approach for Conjunctive Query answering in RSA*. Version 1.1.0. May 2022. URL: `https://doi.org/10.5281/zenodo.6564261`.

[57] Federico Igne, Stefano Germano, and Ian 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*. Ed. by Martin Homola, Vladislav Ryzhikov, and Renate A. Schmidt. Vol. 2954. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: `http://ceur-ws.org/Vol-2954/paper-18.pdf`.

[58] Evgeny Kharlamov, Dag Hovland, Ernesto Jiménez-Ruiz, Davide Lanti, Hallstein Lie, Christoph Pinkel, et al. "Ontology Based Access to Exploration Data at Statoil". In: *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*. Vol. 9367. Lecture Notes in Computer Science. Springer, 2015, pp. 93–112.

[59] Ilianna Kollia and Birte Glimm. "Optimizing SPARQL Query Answering over OWL Ontologies". In: *J. Artif. Intell. Res.* 48 (2013), pp. 253–303. URL: `https://doi.org/10.1613/jair.3872`.

[60] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael 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*. Ed. by Toby Walsh. IJCAI/AAAI, 2011, pp. 2656–2661. URL: `https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-442`.

[61] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael 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.

[62] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. "Conjunctive Queries for a Tractable Fragment of OWL 1.1". In: Jan. 2007, pp. 310–323.

[63] Lee Lacy, Gabriel Aviles, Karen Fraser, William Gerber, Alice Mulvehill, and Robert Gaskill. "Experiences Using OWL in Military Applications." In: Jan. 2005.

[64] Jens Lehmann and Pascal Hitzler. "Concept learning in description logics using refinement operators". In: *Mach. Learn.* 78.1-2 (2010), pp. 203–250. URL: `https://doi.org/10.1007/s10994-009-5146-2`.

[65] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. "The DLV system for knowledge representation and reasoning". In: *ACM Trans. Comput. Log.* 7.3 (2006), pp. 499–562. URL: `https://doi.org/10.1145/1149114.1149117`.

[66] Carsten Lutz. "The Complexity of Conjunctive Query Answering in Expressive Description Logics". In: *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*. Vol. 5195. Lecture Notes in Computer Science. Springer, 2008, pp. 179–193.

[67] Carsten Lutz, Inanç Seylan, David Toman, and Frank Wolter. "The Combined Approach to OBDA: Taming Role Hierarchies Using Filters". In: *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*. Ed. by Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz. Vol. 8218. Lecture Notes in Computer Science. Springer, 2013, pp. 314–330. URL: `https://doi.org/10.1007/978-3-642-41335-3%5C_20`.

[68] Carsten Lutz, David Toman, and Frank 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*. Ed. by Craig Boutilier. 2009, pp. 2070–2075. URL: `http://ijcai.org/Proceedings/09/Papers/341.pdf`.

[69] Bruno Marnette. "Generalized schema-mappings: from termination to tractability". In: *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*. Ed. by Jan Paredaens and Jianwen Su. ACM, 2009, pp. 13–22. URL: `https://doi.org/10.1145/1559795.1559799`.

[70] Brian 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*. Ed. by Stefan Decker, Dieter A. Fensel, Amit P. Sheth, and Steffen Staab. Vol. 40. CEUR Workshop Proceedings. CEUR-WS.org, 2001. URL: `http://CEUR-WS.org/Vol-40/mcbride.pdf`.

[71] Marvin Minsky. "A framework for representing knowledge". In: *Mind Desing*. The MIT Press, 1981.

[72] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. *OWL 2 Web Ontology Language Profiles (Second Edition)*. W3C Recommendation. https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/. W3C, Dec. 2012.

[73] Boris Motik, Yavor Nenov, Robert Piro, and Ian 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*. AAAI Press, 2015, pp. 3127–3133.

[74] Boris Motik, Yavor Nenov, Robert Piro, and Ian Horrocks. "Handling Owl: sameAs via Rewriting". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. AAAI Press, 2015, pp. 231–237.

[75] Boris Motik, Yavor Nenov, Robert Piro, and Ian Horrocks. "Incremental Update of Datalog Materialisation: the Backward/Forward Algorithm". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. AAAI Press, 2015, pp. 1560–1568.

[76] Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, and Dan Olteanu. "Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems". In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. AAAI Press, 2014, pp. 129–137.

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

[78] Boris Motik, Rob Shearer, and Ian Horrocks. “Hypertableau Reasoning for Description Logics”. In: *Journal of Artificial Intelligence Research* 36 (2009), pp. 165–228.

[79] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. “RDFox: A Highly-Scalable RDF Store”. In: *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II.* Vol. 9367. Lecture Notes in Computer Science. Springer, 2015, pp. 3–20.

[80] Giorgio Orsi and Andreas Pieris. “Optimizing Query Answering under Ontological Constraints”. In: *Proc. VLDB Endow.* 4.11 (2011), pp. 1004–1015. URL: http://www.vldb.org/pvldb/vol4/p1004-orsi.pdf.

[81] Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. “Data Complexity of Query Answering in Expressive Description Logics via Tableaux”. In: *J. Autom. Reason.* 41.1 (2008), pp. 61–98. URL: https://doi.org/10.1007/s10817-008-9102-9.

[82] Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus. “Query Answering in the Horn Fragments of the Description Logics SHOIQ and SROIQ”. In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011.* IJCAI/AAAI, 2011, pp. 1039–1044.

[83] Magdalena Ortiz, Sebastian Rudolph, and Mantas Simkus. “Worst-Case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2”. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010.* Ed. by Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski. AAAI Press, 2010. URL: http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1296.

[84] Magdalena Ortiz and Mantas Simkus. “Reasoning and Query Answering in Description Logics”. In: *Reasoning Web. Semantic Technologies for Advanced Query Answering - 8th International Summer School 2012, Vienna, Austria, September 3-8, 2012. Proceedings.* Ed. by Thomas Eiter and Thomas Krennwallner. Vol. 7487. Lecture Notes in Computer Science. Springer, 2012, pp. 1–53. URL: https://doi.org/10.1007/978-3-642-33158-9%5C_1.

[85] David Osumi-Sutherland, Simon Reeve, Christopher Mungall, Fabian Neuhaus, Alan Ruttenberg, Gregory Jefferis, and James Armstrong. “A strategy for building neuroanatomy ontologies”. In: *Bioinformatics (Oxford, England)* 28 (Mar. 2012), pp. 1262–9.

[86] Jeff Z. Pan and Edward Thomas. “Approximating OWL-DL Ontologies”. In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada.* AAAI Press, 2007, pp. 1434–1439. URL: http://www.aaai.org/Library/AAAI/2007/aaai07-227.php.

[87] Héctor Pérez-Urbina, Ian Horrocks, and Boris 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.* Vol. 5823. Lecture Notes in Computer Science. Springer, 2009, pp. 489–504.

[88] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. "Tractable query answering and rewriting under description logic constraints". In: *J. Appl. Log.* 8.2 (2010), pp. 186–209. URL: https://doi.org/10.1016/j.jal.2009.09.004.

[89] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. "Linking Data to Ontologies". In: *J. Data Semant.* 10 (2008), pp. 133–173. URL: https://doi.org/10.1007/978-3-540-77688-8%5C_5.

[90] Ross Quillian. "Semantic Memory". In: (1968). Ed. by Marvin Minsky, pp. 216–270.

[91] Yuan Ren, Gerd Gröner, Jens Lemcke, Tirdad Rahmani, Andreas Friesen, Yuting Zhao, Jeff Z. Pan, and Steffen Staab. "Validating Process Refinement with Ontologies". In: *Proceedings of the 22nd International Workshop on Description Logics (DL 2009), Oxford, UK, July 27-30, 2009.* Ed. by Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler. Vol. 477. CEUR Workshop Proceedings. CEUR-WS.org. URL: http://ceur-ws.org/Vol-477/paper%5C_59.pdf.

[92] Yuan Ren, Jeff Z. Pan, Isa Guclu, and Martin 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.* Vol. 9898. Lecture Notes in Computer Science. Springer, 2016, pp. 167–183.

[93] Peter Nick Robinson and Sebastian Bauer. "Introduction to Bio-Ontologies". In: 2011.

[94] Riccardo 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.* Ed. by Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris. Vol. 250. CEUR Workshop Proceedings. CEUR-WS.org, 2007. URL: http://ceur-ws.org/Vol-250/paper%5C_83.pdf.

[95] Sebastian Rudolph and Birte Glimm. "Nominals, Inverses, Counting, and Conjunctive Queries or: Why Infinity is your Friend!" In: *J. Artif. Intell. Res.* 39 (2010), pp. 429–481. URL: https://doi.org/10.1613/jair.3029.

[96] Viachaslau Sazonau and Uli Sattler. "Mining Hypotheses from Data in OWL: Advanced Evaluation and Complete Construction". In: *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I.* Ed. by Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin. Vol. 10587. Lecture Notes in Computer Science. Springer, 2017, pp. 577–593. URL: https://doi.org/10.1007/978-3-319-68288-4%5C_34.

[97] Andrea Schaerf. "On the Complexity of the Instance Checking Problem in Concept Languages with Existential Quantification". In: *J. Intell. Inf. Syst.* 2.3 (1993), pp. 265–278. URL: https://doi.org/10.1007/BF00962071.

[98] Guus Schreiber and Yves Raimond. *RDF 1.1 Primer.* W3C Note. https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/. W3C, June 2014.

[99] Bart Selman and Henry A. Kautz. “Knowledge Compilation and Theory Approximation”. In: *J. ACM* 43.2 (1996), pp. 193–224. URL: https://doi.org/10.1145/226643.226644.

[100] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. “Pellet: A practical OWL-DL reasoner”. In: *J. Web Semant.* 5.2 (2007), pp. 51–53. URL: https://doi.org/10.1016/j.websem.2007.03.004.

[101] Martin Skjæveland, Espen Lian, and Ian Horrocks. “Publishing the Norwegian Petroleum Directorate’s FactPages as Semantic Web Data”. In: Oct. 2013, pp. 162–177.

[102] Martin G. Skjæveland, Henrik Forssell, Johan W. Klüwer, Daniel P. Lupp, Evgenij Thorstensen, and Arild Waaler. “Pattern-Based Ontology Design and Instantiation with Reasonable Ontology Templates”. In: *Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017*. Ed. by Eva Blomqvist, Óscar Corcho, Matthew Horridge, David Carral, and Rinke Hoekstra. Vol. 2043. CEUR Workshop Proceedings. CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-2043/paper-04.pdf.

[103] Martin G. Skjæveland, Henrik Forssell, Johan W. Klüwer, Daniel P. Lupp, Evgenij Thorstensen, and Arild Waaler. “Reasonable Ontology Templates: APIs for OWL”. In: *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*. Ed. by Nadeschda Nikitina, Dezhao Song, Achille Fokoue, and Peter Haase. Vol. 1963. CEUR Workshop Proceedings. CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1963/paper597.pdf.

[104] Martin G. Skjæveland, Daniel P. Lupp, Leif Harald Karlsen, and Johan W. Klüwer. “OTTR: Formal Templates for Pattern-Based Ontology Engineering”. In: *Advances in Pattern-Based Ontology Engineering, extended versions of the papers published at the Workshop on Ontology Design and Patterns (WOP)*. Ed. by Eva Blomqvist, Torsten Hahmann, Karl Hammar, Pascal Hitzler, Rinke Hoekstra, Raghava Mutharaju, et al. Vol. 51. Studies on the Semantic Web. IOS Press, 2021, pp. 349–377. URL: https://doi.org/10.3233/SSW210025.

[105] Giorgio Stefanoni and Boris Motik. “Answering Conjunctive Queries over EL Knowledge Bases with Transitive and Reflexive Roles”. In: *CoRR* abs/1411.2516 (2014).

[106] Giorgio Stefanoni and Boris Motik. “Answering Conjunctive Queries over EL Knowledge Bases with Transitive and Reflexive Roles”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 1611–1617. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9310.

[107] Giorgio Stefanoni, Boris Motik, and Ian Horrocks. “Introducing Nominals to the Combined Query Answering Approaches for EL”. In: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. Ed. by Marie desJardins and Michael L. Littman.

AAAI Press, 2013. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6156.

[108] Andreas Steigmiller and Birte 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.* Ed. by Mantas Simkus and Grant E. Weddell. Vol. 2373. CEUR Workshop Proceedings. CEUR-WS.org, 2019. URL: http://ceur-ws.org/Vol-2373/paper-25.pdf.

[109] Andreas Steigmiller and Birte Glimm. "Parallelised ABox Reasoning and Query Answering with Expressive Description Logics". In: *The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings*. Ed. by Ruben Verborgh, Katja Hose, Heiko Paulheim, Pierre-Antoine Champin, Maria Maleshkova, Óscar Corcho, Petar Ristoski, and Mehwish Alam. Vol. 12731. Lecture Notes in Computer Science. Springer, 2021, pp. 23–39. URL: https://doi.org/10.1007/978-3-030-77385-4%5C_2.

[110] Andreas Steigmiller, Birte Glimm, and Thorsten Liebig. "Reasoning with Nominal Schemas through Absorption". In: *J. Autom. Reason.* 53.4 (2014), pp. 351–405. URL: https://doi.org/10.1007/s10817-014-9310-4.

[111] Andreas Steigmiller, Thorsten Liebig, and Birte Glimm. "Konclude: System Description". In: *Journal of Web Semantics (JWS)* 27 (2014), pp. 78–85.

[112] Giorgos Stoilos. "Hydrowl: A Hybrid Query Answering System for OWL 2 DL Ontologies". In: *Web Reasoning and Rule Systems - 8th International Conference, RR 2014, Athens, Greece, September 15-17, 2014. Proceedings*. Ed. by Roman Kontchakov and Marie-Laure Mugnier. Vol. 8741. Lecture Notes in Computer Science. Springer, 2014, pp. 230–238. URL: https://doi.org/10.1007/978-3-319-11113-1%5C_20.

[113] Giorgos 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, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*. Ed. by Valentina Presutti, Claudia d'Amato, Fabien Gandon, Mathieu d'Aquin, Steffen Staab, and Anna Tordai. Vol. 8465. Lecture Notes in Computer Science. Springer, 2014, pp. 317–332. URL: https://doi.org/10.1007/978-3-319-07443-6%5C_22.

[114] Edward Thomas, Jeff Z. Pan, and Yuan Ren. "TrOWL: Tractable OWL 2 Reasoning Infrastructure". In: *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II*. Ed. by Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache. Vol. 6089. Lecture Notes in Computer Science. Springer, 2010, pp. 431–435. URL: https://doi.org/10.1007/978-3-642-13489-0%5C_38.

[115] Alvaro del Val. "First order LUB approximations: characterization and algorithms". In: *Artif. Intell.* 162.1-2 (2005), pp. 7–48. URL: https://doi.org/10.1016/j.artint.2004.01.003.

[116] Johanna Völker and Mathias Niepert. “Statistical Schema Induction”. In: *The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings, Part I.* Ed. by Grigoris Antoniou, Marko Grobelnik, Elena Paslaru Bontas Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Z. Pan. Vol. 6643. Lecture Notes in Computer Science. Springer, 2011, pp. 124–138. URL: `https://doi.org/10.1007/978-3-642-21034-1%5C_9`.

[117] Sebastian Wandelt, Ralf Möller, and Michael Wessel. “Towards Scalable Instance Retrieval over Ontologies”. In: *Int. J. Softw. Informatics* 4.3 (2010), pp. 201–218. URL: `http://www.ijsi.org/ch/reader/view%5C_abstract.aspx?file%5C_no=i59`.

[118] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. “Ontology-Based Data Access: A Survey”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.* ijcai.org, 2018, pp. 5511–5519.

[119] Yujiao Zhou. “Pay-As-You-Go Ontology Query Answering Using a Datalog Reasoner”. PhD thesis. University of Oxford, 2015.

[120] Yujiao Zhou, Bernardo Cuenca Grau, Yavor Nenov, Mark Kaminski, and Ian Horrocks. “PAGOdA: Pay-As-You-Go Ontology Query Answering Using a Datalog Reasoner”. In: *Journal of Artificial Intelligence Research* 54 (2015), pp. 309–367.

Index