SAPIENZA
UNIVERSITÀ DI ROMA

Privacy-preserving and Inconsistency-tolerant Query Answering in Knowledge Bases and Databases

Department of Computer, Control, and Management Engineering Antonio Ruberti

Ph.D. in Engineering of Computer Science (XXXVI cycle)

Lorenzo Marconi
ID number 1494505

Advisor
Prof. Riccardo Rosati

Co-Advisor
Prof. Leonardo Querzoni

Academic Year 2023/2024

Thesis defended on May 7th, 2024
in front of a Board of Examiners composed by:

Prof. Valeria Cardellini (Università degli Studi di Tor Vergata)
Prof. Giuliana Vitiello (Università degli Studi di Salerno)
Prof. Alberto Pretto (Università degli Studi di Padova)

Thesis reviewed by:

Prof. Leopoldo Bertossi (Carleton University, Ottawa, Canada)
Prof. Thomas Lukasiewicz (University of Oxford, United Kingdom)

Privacy-preserving and Inconsistency-tolerant Query Answering in Knowledge Bases and Databases
PhD thesis. Sapienza University of Rome

© 2024 Lorenzo Marconi. All rights reserved

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: marconi@diag.uniroma1.it

To my awesome family,
I can't express how much I feel lucky to rely on your unbounded support and love. Thank you for constantly bridging the physical distance between us with your affection.
To Riccardo, my esteemed supervisor,
I feel I have learned a lot during my doctoral experience, and this is largely thanks to your guidance. I sincerely appreciate all the time you dedicated to me; I never took it for granted.
To my colleagues,
The collaborative spirit that I found in our research group is truly invaluable, and the kind and stimulating atmosphere that you set up made these past years so much more enjoyable.
To Dorian, my best man and best friend,
Thank you for accompanying me during these years. I deeply value our solid and special bond and, no matter what's going on, I know I can count on you to be a steady presence in my life. See you on the plane!
And to Luana, my beloved girlfriend,
who honored me becoming my wife in the middle of this journey,
You have a gift for understanding my unspoken needs, and I'm so grateful for all the big and little things that you do for me every day. Thank you for always finding the perfect way to make me feel better. Just like all other achievements of mine, this one belongs to you as well!

Abstract

In recent decades, the widespread use of information systems has led to an increasing amount of sensitive data managed, posing several privacy and security risks. *Controlled Query Evaluation* (*CQE*) is a framework for limiting access to sensitive information when querying a system, while still providing useful responses. Initially proposed for databases, CQE has recently been studied for Description Logics ontologies, the logical formalization underpinning most Semantic Web technologies.

Central to CQE is the possibility of specifying a data protection *policy* in a declarative way, avoiding the need to actively alter data to ensure their security. The policy is enforced via *optimal censors*, which aim to minimally alter information while maintaining confidentiality. Notably, censors satisfying the so-called *indistinguishability* property also ensure that the end user is not even able to discern if the answers to a query have been altered or not.

The thesis investigates various CQE semantics based on the so-called *optimal GA censors*, which enjoy the indistinguishability property. In order not to arbitrarly pick a censor, we adopt the usual strategy of *skeptical reasoning*, i.e. answering according to all the optimal censors. Unfortunately, this approach is proven to be intractable, so we present an alternative semantics based on the intersection of all optimal GA censors (called *IGA censor*). Under this semantics, answering union of conjunctive queries (UCQs) in the Description Logic DL-Lite$_{\mathcal{R}}$ is first-order (FO) rewritable, and thus in AC^0 in data complexity.

Anyway, the intersection-based approach may cause losing too much information, which led us to explore smarter semantics for improving the throughput of query answers while preserving confidentiality. Towards this aim, we first show how to exploit a priority relation between ontology predicates for reducing the number of censors. Then, we consider the strategy of selecting censors dynamically, exploiting the order in which queries are posed for maximizing the *cooperativity* of the system. In both cases, we provide suitable FO rewriting techniques, proving that the nice computational properties of the IGA semantics are preserved.

The prioritized scenario also served as main setting for carrying out our experiments, wherein we implemented the newly-presented CQE techniques adapting them to the ontology-based data access methodology.

Moreover, the research extends to the related field of *Consistent Query Answering* (*CQA*), which studies how to handle inconsistencies when answering queries. Specifically, we consider knowledge bases in which a set of existential rules must always be satisfied by the underlying database. A central notion in CQA is the one of *repair*, that is a maximal subset of the database satisfying all the rules. As for query answering, similarly as done in CQE, we examine both skeptical reasoning and the intersection-based semantics, known as *AR* and *IAR*, respectively. We study a very expressive language of rules and identify many subclasses where repair checking and UCQ entailment are tractable, or even FO rewritable.

We finally investigate the integration between *open* and *closed-world assumption*, offering insights into utilizing open and closed predicates within the CQA framework, and provide the first set of complexity results for the aforenamed decision problems within this intriguing scenario.

Publications

This dissertation presents the research conducted throughout my three-year doctoral program, and part of its content has already been featured in the subsequent scientific publications:

[1] Bonatti, P., Cima, G., Lembo, D., Marconi, L., Rosati, R., and Savo, D. Enhancing cooperativity in controlled query evaluation over ontologies. *Artificial Intelligence*. Submitted.

[2] Cima, G., Lembo, D., Marconi, L., Rosati, R., and Savo, D. Indistinguishability in controlled query evaluation over prioritized description logic ontologies. *J. of Web Semantics*. Submitted.

[3] Cima, G., Lembo, D., Marconi, L., Rosati, R., and Savo, D. Enhancing controlled query evaluation through epistemic policies (2024). Accepted to the 33rd Int. Joint Conf. on Artificial Intelligence (IJCAI).

[4] Cima, G., Lembo, D., Marconi, L., Rosati, R., and Savo, D. A gentle introduction to controlled query evaluation in DL-Lite ontologies. *SN Computer Science*, **5** (2024).

[5] Bonatti, P., Cima, G., Lembo, D., Marconi, L., Rosati, R., Sauro, L., and Savo, D. Dynamic controlled query evaluation over DL-Lite ontologies (extended abstract). In *Proc. of the 36th Int. Workshop on Description Logic (DL)*, vol. 3515 of *CEUR Electronic Workshop Proc.,* `http://ceur-ws.org/` (2023).

[6] Bonatti, P., Cima, G., Lembo, D., Marconi, L., Rosati, R., Sauro, L., and Savo, D. Controlled query evaluation in OWL 2 QL: A "longest honeymoon" approach. In *Proc. of the 21st Int. Semantic Web Conf. (ISWC)*, vol. 13489 of *Lecture Notes in Computer Science*, pp. 428–444. Springer (2022).

[7] Cima, G., Lembo, D., Marconi, L., Rosati, R., Savo, D., and Sinibaldi, D. Controlled query evaluation over ontologies through policies with numerical restrictions. In *Proc. of the 4th IEEE Int. Conf. on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 33–36. IEEE (2021).

[8] Cima, G., Lembo, D., Marconi, L., Rosati, R., and Savo, D. Controlled query evaluation over prioritized ontologies with expressive data protection policies. In *Proc. of the 20th Int. Semantic Web Conf. (ISWC)*, vol. 12922 of *Lecture Notes in Computer Science*, pp. 374–391. Springer (2021).

[9] Cima, G., Lembo, D., Marconi, L., Rosati, R., and Savo, D. On information disclosure in ontology-based data access (extended abstract). In *Proc. of the 34th Int. Workshop on Description Logic (DL)*, vol. 2954 of *CEUR Electronic Workshop Proc.,* `http://ceur-ws.org/` (2021).

[10] Cima, G., Lembo, D., Marconi, L., Rosati, R., and Savo, D. Controlled query evaluation in ontology-based data access. In *Proc. of the 19th Int. Semantic Web Conf. (ISWC)*, vol. 12506 of *Lecture Notes in Computer Science*, pp. 128–146. Springer (2020).

Contents

1 Introduction **1**
1.1 Contributions 7
1.2 Thesis structure overview 9

2 Preliminaries **11**
2.1 First-order logic 11
2.2 Notes on complexity theory 13
2.3 Description logics 13
2.3.1 Syntax of a DL-Lite$_{\mathcal{R}}$ ontology 14
2.3.2 Semantics of a DL-Lite$_{\mathcal{R}}$ ontology 15
2.4 Query answering and FO-rewritability 16
2.5 Ontology-Based Data Access 17

3 Controlled query evaluation for DL ontologies **21**
3.1 Framework 21
3.2 Query answering under optimal GA censors 26
3.3 IGA censor 28
3.3.1 Query answering under IGA censor 30

4 Prioritized controlled query evaluation **37**
4.1 Framework extension and common priority-based semantics 37
4.2 Deterministic priority-based censors 40
4.3 Query answering under k-DD censor 43

5 Extending ontology and policy languages **47**
5.1 DL-Lite$_{\mathcal{A}}$ ontologies and policies with comparison atoms 47
5.2 DL-Lite$^{\mathcal{H}}_{horn}$ ontologies and policies with number restrictions 50

6 Practical implementation and experiments **53**
6.1 Policy-protected ontology-based data access 53
6.2 Inexpressibility results 56
6.3 Embedding a policy into the mapping 57
6.4 Experiments 66

7 Dynamic controlled query evaluation **71**
7.1 Dynamic framework and the longest honeymoon property 71
7.2 First-order rewritability of BUCQ entailment 78

7.3 Towards practical techniques and approximations 84

8 Consistent query answering 87
8.1 Dependencies 87
8.1.1 Subclasses of dependencies 88
8.1.2 The chase 89
8.2 Repairs and decision problems 89
8.3 Recoverability 91
8.4 Repair checking 95
8.5 IAR-entailment 101
8.6 AR-entailment 107

9 Consistent query answering with open and closed predicates 111
9.1 CQA semantics with open and closed predicates 111
9.2 Open-closed-separable dependencies 113
9.2.1 OC-entailment 114
9.2.2 Repair checking 116
9.2.3 OC-AR-entailment 116
9.2.4 OC-IAR-entailment 117

10 Related work 119
10.1 Controlled query evaluation 119
10.2 Consistent query answering 122

11 Conclusions 125

Bibliography 129

A The *PerfectRef* algorithm 139

B Complexity of decision problems in propositional logic 141
B.1 Introduction to propositional logic 141
B.2 Complexity of satisfiability and validity problems 142
B.2.1 CNF-SAT 142
B.2.2 3-SAT 142
B.2.3 Horn-CNF-SAT 143
B.2.4 Horn-3-SAT 143
B.2.5 2-QBF$^{\forall}$ 143

Alphabetical index 145

Chapter 1

Introduction

In the last decades, the use of information systems in our daily life has grown significantly and it now pervades several aspects of modern society. A substantial part of the knowledge managed by such systems consists in sensitive data, including medical records, financial details, and data utilized for web profiling, among others. Without adequate protection, the exposure of such private information could jeopardize the privacy and security of the individuals involved [54].

The risk of data breaches is often faced through a range of strategies, possibly combined together, among which: "physical" defense methods, including firewalls and intrusion detection systems [107]; access control systems [36], which allow accessing to sensitive data only to authenticated users with appropriate privileges and/or satisfying specific conditions; data perturbation by means of encryption [23] or anonymization [14, 59] techniques; data filtering through the exposure of partial views of the original dataset [121].

The goal of *privacy-preserving query answering* [55] is to prevent the end user from gathering confidential data while still providing them useful answers whenever possible. *Controlled Query Evaluation* (*CQE*) is a framework designed for addressing this issue by refusing to respond to certain queries or altering answers when necessary to protect sensitive knowledge. CQE was introduced in the realm of databases [108, 29, 31] and more recently gained attention in the context of Description Logics (DL) ontologies [37, 58, 83]. DLs provide the essential logic behind the Semantic Web [20], empowering machines to process information with human-like understanding, leading to smarter search, personalized Web experiences, and seamless data integration within cross-domain applications. With this powerful tool, defining a secure CQE framework is even more challenging, since we deal with the possibility of inferring implicit knowledge from a starting ground truth, which significantly raises the risk of inadvertent leakage of confidential information.

A distinctive feature of the proposed approach is the capability to prescribe in a declarative way the protection *policy*, i.e. a set of logic-based rules that the query answering system should adhere to: such rules implicitly define the information that should be kept undisclosed from the user. Thus, the individual responsible for data security[1] does not necessarily have to know *how* the desired behavior can be implemented, i.e. the underlying technical mechanisms, but she just need to identify

[1] A role that Articles 37–39 of GDPR [66] recently codified as Data Protection Officer (DPO).

and declare *what* qualifies a secret as such.

Example 1. Some disease may be highly correlated with specific age groups and symptoms. For instance, older individuals (typically over 65) suffering of dysuria may be affected of prostate enlargement, medically referred to as benign prostatic hyperplasia (BPH). Therefore, if it is known that someone over 65 has purchased an antidysuric medicine, it could suggest that they are affected by BPH. To counter such an assumption, we define the following rule.

$$\forall p, a, m\,(\mathsf{age}(p, a) \wedge a > 65 \wedge \mathsf{buy}(p, m) \wedge \mathsf{Antidysuric}(m) \rightarrow \bot).$$

▽

An important aspect that must be taken into account when dealing with security is the set of assumptions made on the attacker (in our scenario, any possible malicious user trying to gain access to confidential information). In order not to employ the bad practice of *security by obscurity*, our examination follows the attack model described in [35], i.e. we assume that the end user is aware of the specified protection policy, the algorithm applied for enforcing it, and also the set of all the intensional rules (known as *TBox*) used to infer new knowledge; on the other hand, we assume that she originally does not know any fact belonging to the ground truth (known as *ABox*), nor any logical consequences thereof.

In practice, to enforce the data protection policy, query answering is altered by a function called *censor*. The information disclosable by a censor consists in a set of logical formulas expressed in some first-order (FO) language. The two main families of censors presented are the GA and CQ censors. The first ones are mathematical functions taking as input an ABox and whose values are sets of *ground atoms*[2], i.e. atomic formulas without variables. The latter return instead sets of *Boolean conjunctive queries* (*BCQs*), i.e. positive conjunctions of predicate atoms where every variable is existentially quantified. Among possible censors, *optimal* ones are preferred, namely, those altering query answers in a minimal way (with respect to set inclusion).

Also, this work places specific emphasis on the so-called notion of *indistinguishability* [34, 37, 52, 35]. In a nutshell, such a property guarantees that the end user is not capable of distinguishing if any query answer has been altered or not, because there always exists a data instance that complies with the policy and, at the same time, serves as support for all the provided answers. This fundamental property drives us to focus on optimal GA censors. In fact, we demonstrate that such kind of censors enjoy the property of indistinguishability.

The CQE framework and all the formal properties for censors are given for any DL. However, specific techniques conceived for solving the examined problems and also the provided complexity results are valid for various logics belonging to the DL-Lite family of DLs, which are well-known for their valuable computational properties and their scalability with respect to the amount of considered data. A key member of such a family is DL-Lite$_{\mathcal{R}}$, which is well-known for underpinning OWL 2 QL, one of the three profiles of OWL 2 [56, 97], the standard ontology language recommended by W3C for Semantic Web applications. A distinguishing

[2]We say that the function *returns* a set of ground atoms.

feature of the above logic is that answering to union of conjunctive queries (UCQs) over DL-Lite$_\mathcal{R}$ ontologies is FO rewritable, which means that it can be reduced to the evaluation of an SQL query over a database.

As for the adopted CQE semantics, first we mention the strategy of arbitrarily choosing an optimal GA censor and use its materialization as the new, policy-safe ABox component of the ontology accessible by the user. Despite offering a tractable solution, we argue that this approach is not coherent with the declarative nature of CQE. We hence analyze the case of *skeptical entailment* of UCQs under all the optimal GA censors, which lets us avoid the previously-mentioned arbitrary selection process. However, it turns out that the computational cost of query answering in this setting is problematic: in particular, the data complexity of query answering over DL-Lite$_\mathcal{R}$ ontologies is intractable (coNP-complete).

To address the above computational issue, we propose an alternative semantics centered around the notion of *intersection of all the optimal GA censors*, which we call *IGA censor*. By providing a suitable FO-rewriting technique, we show that query answering over DL-Lite$_\mathcal{R}$ ontologies under the new intersection-based semantics for CQE has the same data complexity as under the standard semantics, i.e. it is in AC^0 in data complexity. This result indicates the possibility of effective realization of practical CQE systems based on the notion of IGA censor. This approach, however, shows limitations with respect to the maximization of the information disclosed through query answering.

In order to improve the above aspect, we add to the CQE framework the possibility of specifying a priority relation between ontology predicates. This priority relation is used for restricting the set of admissible censors and, thus, augmenting the set of query answers both under the skeptical-entailment semantics and under the intersection-based semantics. While preferences increase the amount of disclosable information, their use constitutes in general an overhead in terms of computational cost of query answering, as already shown e.g. in [25]. However, we are able to identify an intersection-based semantics for CQE with preferences under which answering UCQs over DL-Lite$_\mathcal{R}$ ontologies is tractable in data complexity and not harder than in the absence of preferences. In particular, under this semantics for CQE, query answering is in AC^0 in data complexity and enjoys the FO-rewritability property. This means that such an approach to CQE is potentially able to increase confidentiality-preserving answers to queries while remaining feasible in practice.

A further important property that we consider is the *cooperativity* of the query-answering system, which should preferably take into account the users' interests (specifically, the history of past queries) when selecting the censoring strategy. Taking inspiration from the research of Biskup and Bonatti [32], we thus introduce the notion of *maximally cooperative* censors, which are dynamically selected based on the stream of user queries. Intuitively, for each query posed, we aim to answer honestly as long as there exists a censor allowing us to do so. This approach, known as "longest honeymoon" [32], results to be very natural: in situations where specific details about users' intentions are missing, we can assume that the order in which queries are submitted indicates their relative significance. Moreover, given our lack of foresight regarding future queries, if we can respond truthfully to the current one while complying with the data protection policy, we should seize the opportunity to maximize our level of cooperation.

The good computational properties of DL-Lite$_\mathcal{R}$ make the choice of such a logic best-suited for *Ontology-based Data Access* (*OBDA*) applications. OBDA [100] is a methodology that involves enriching one or more data sources, possibly consisting of heterogeneous and unstructured data, by means of an ontology. The role of the ontology, more specifically of the TBox, is to depict a particular domain of interest at a more abstract level, regardless on how the data is actually stored and maintained within the system's data layer. The connection between the ontology and the underlying data is established by adding to the OBDA instance a third, intermediary layer called *mapping*.

In this thesis we propose a fresh perspective on CQE, employing an OBDA-embedded methodology: given as input an OBDA instance and a protection policy, we aim to build a new OBDA instance that retains the same data and ontology layers but incorporates a new mapping, which accounts for all the rules specified in the policy. The newly created mapping can then be seamlessly integrated into off-the-shelf tools for OBDA without requiring any software modifications for guaranteeing confidentiality preservation. In this way, we ensure that querying the new OBDA instance accurately reflects the CQE approach described above.

Furthermore, CQE is known [83] to have close ties to *Consistent Query Answering* (*CQA*). The main goal of CQA is to answer queries in an *inconsistency-tolerant* manner, i.e. mimicking the scenario where answers derive from a knowledge base (KB) free of inconsistencies. This makes query answers more accurate, reliable, and hence suitable for data analysis and decision-making purposes. The connection between the two frameworks lies in the intuition that ensuring an ontology adheres to a data protection policy, as expressed through logical formulas, is analogous to verifying the coherence of the logical theory formed by both the ontology and the policy. Thus, beyond their inherent interest, techniques addressing CQA problems can often be highly useful in the context of data protection. Such a close relationship motivated us to also explore this further research topic.

First introduced in 1999 by Arenas *et al.* [7], CQA has been deeply studied by the scientific community [50, 21, 119]. The central notion of consistent query answering is the one of *repair*. Given an *inconsistent* database or KB $\mathcal{D}$, a repair of $\mathcal{D}$ is a consistent database or KB that is "maximally close" to $\mathcal{D}$. Several distinct formalizations of such notion of maximality and inconsistency are possible, giving rise to different types of repairs. In general, many different repairs for the same database or KB may exist. The problem of consistent query answering as originally formulated was: given a query q, compute the answers to q that are true in every repair. Also the *repair checking* problem is often accounted, i.e. the problem of deciding whether a candidate $\mathcal{D}'$ is a repair of the original database or KB $\mathcal{D}$.

CQA was originally defined for relational databases, but it has then been studied also in the context of Description Logics ontologies (see e.g. [105, 24, 27]) and rule-based KBs, in particular knowledge bases with *existential rules* [88, 39], where an extensional database is coupled with a set of first-order implications. Our research on CQA falls within this latest scenario.

We consider a very expressive class of existential rules that comprises and extends both disjunctive tuple-generating dependencies (DTGDs) and equality-generating dependencies (EGDs), which are the most studied forms of rules studied in the literature. Specifically, the rules that we consider in this thesis correspond to

the so-called *disjunctive embedded dependencies with inequalities* ($DED^{\neq}s$) [63, 8, 62]. We also consider five subclasses of these dependencies, corresponding to some of the most important ones studied in the literature on existential rules, i.e. the classes of *acyclic*, *full*, *guarded*, *linear*, and *sticky* rules [40, 12, 41].

Concerning the semantics of consistent query answering, we adopt a *closed* interpretation of the predicates, and the well-known *tuple-deletion* semantics for repairs [51, 118, 2, 114], according to which a repair is a maximal (with respect to set inclusion) subset of the initial database that satisfies the rules (comparing the notion of repair with the one of censor as seen in CQE arises now spontaneously).

Example 2. Let D be a quaternary relational table representing that a patient in a hospital is suffering from some disease. It takes as arguments an identifier for internal use plus the patient's name, age and disease. The data protection officer considers the projection of D over name and age to be a *quasi-identifier*, i.e. something that is almost sufficient for identifying a person, and she wants to guarantee the *k-anonymity* [113] of the knowledge base for a factor $k = 5$.

This task can be easily accomplished in the CQA framework. For instance, the system administrator could specify the following rule:

$$\forall p_1, n, a, d_1 \Big(D(p_1, n, a, d_1) \rightarrow \exists p_2, d_2, \ldots, p_5, d_5 \Big(D(p_2, n, a, d_2) \wedge \cdots \wedge D(p_5, n, a, d_5) \wedge \bigwedge_{1 \leq i < j \leq 5} p_i \neq p_j \Big) \Big)$$

In a tuple-deletion scenario, enforcing this rule implies showing only the tuples of D for which there exist at least four other tuples having the same value for name and age.

We observe that other kinds of constraints such as TGDs or EGDs do not generally allow for inequalities in the head, and that such inequalities can not be simulated by the addition of equalities in the body. ▽

Given these premises, we analyze the aforenamed decision problems, that is, the repair checking and query entailment problems. For the latter, we still consider the language of UCQs, and analyze two different inconsistency-tolerant semantics: *AR-entailment*, i.e. skeptical entailment over all the repairs, and *IAR-entailment*, i.e. entailment over the intersection of all the repairs. Notably, IAR-entailment corresponds to a sound approximation of AR-entailment.

We study the complexity of the above problems, providing a comprehensive analysis of the data complexity (i.e. the complexity with respect to the size of the database) of repair checking and both AR-entailment and IAR-entailment of UCQs, for all the possible combinations of the five classes of existential rules considered in this thesis. The results are summarized in Table 8.1. In particular, despite consistent query answering is well-known to be a computationally hard task, we identify several cases in which the above problems are tractable, or even FO rewritable, and present new query rewriting techniques that can be the basis for practical inconsistency-tolerant query answering systems.

The closest work to our research on CQA is authored by Lukasiewicz *et al.* [88], which present an exhaustive study of the complexity of consistent query answering over existential rules. That paper inherits the open-world assumption (OWA)

adopted by the recent approaches to existential rules, in which the interpretation of all predicates is open, thus repairing an inconsistency between the database and the rules is preferably done by (virtually) adding tuples to the initial database (that is, tuple deletions are considered only if tuple additions can not repair the data). Conversely, our approach adopts a closed-world assumption (CWA), in which all predicates are closed and the only way to repair an inconsistency is through tuple deletions. Given this semantic discrepancy, the two problems studied are actually different, so our findings can be seen as complementary to the ones presented in the aforementioned work (see also the conclusions). We also remark that both the language for existential rules and the query language considered there are less expressive than the ones (disjunctive dependencies and UCQs) studied in this thesis.

Many real-life situations, however, involve grappling with partially incomplete information. This may occur both when certain data is actually not available (or part of it must remain undisclosed, as in CQE) or when, by design choice, additional knowledge can be inferred from an underlying, complete dataset. Modeling this scenario is often achieved by specifying which predicates must be interpreted in a "closed" (i.e. complete) or "open" (i.e. incomplete) way [95]. Handling partially-complete knowledge has a pivotal importance also in the realm of data protection. As well-explained in [37, 35], there are several situations where, if an attacker is aware that the system has complete knowledge over part of the domain, she can exploit this form of *meta-knowledge* to employ non-standard forms of reasoning for inferring the secret.

Therefore, driven by our findings for the case of CWA and the analogous drawn for OWA in [88], in this thesis we also study how to use open and closed predicates in the CQA framework. In particular, we identify a suitable subclass of DED$^{\neq}$s, named *open-closed separable* (*OCS*) dependencies, for which we are able to provide a first set of complexity results (illustrated in Table 9.1) both for repair checking and entailment of UCQs under AR and IAR semantics.

A possible way of dealing with inconsistencies in a scenario with open and closed predicates is exemplified below.

Example 3. A university department has complete knowledge about scholarships and exams, which are modeled with, respectively, a unary predicate S taking just the student as argument, and a ternary predicate E accepting three arguments, namely the student, course and year of the exam. Each scholarship may be merit-based (M) or a grant for foreign students (G). Moreover, the department may have optional (hence open) knowledge about the students' nationality and income. Specifically, a student may be a resident student (R) or a student coming from a foreign country (F), but not both, and may or may not have a low income (L). A necessary condition for having a grant or a merit-based scholarship is, respectively, being a foreign student with a low income or having passed three exams in the same year. Such a domain can be modeled by specifying the following rules:

$$\begin{aligned}
&\forall s\,(S(s) \rightarrow M(s) \vee G(s))\\
&\forall s, c\,(R(s) \wedge F(s,c) \rightarrow \bot)\\
&\forall s\,(G(s) \rightarrow \exists c\,(F(s,c) \wedge L(s)))\\
&\forall s\,(M(s) \rightarrow \exists c_1, c_2, c_3, y\,(E(s,c_1,y) \wedge E(s,c_2,y) \wedge E(s,c_3,y) \wedge\\
&\qquad\qquad c_1 \neq c_2 \wedge c_1 \neq c_3 \wedge c_2 \neq c_3))
\end{aligned}$$

Unfortunately, the school's database is not compliant with some of these rules. In fact, we know that the following facts hold:

$$G(\mathsf{ann}), G(\mathsf{sam}), R(\mathsf{sam}), M(\mathsf{tom}), E(\mathsf{tom}, \mathsf{nlp}, 2023), E(\mathsf{tom}, \mathsf{kr}, 2023)$$

In words, both Ann and Sam have a grant but we are not explicitly told that they are foreign students. Since F is an open predicate, we can infer that Ann is a foreign student, but we can not do the same for Sam, because we know that he is a resident student. Therefore, we are constrained to either remove the fact $G(\mathsf{sam})$ or $R(\mathsf{sam})$ from the knowledge base.

For what concerns the last rule, since the information about exams is complete, we can not suppose that there exists an unknown third exam which was passed by Tom, thus the only possibility of repairing this second inconsistency implies removing $M(\mathsf{tom})$ from the knowledge base. ▽

1.1 Contributions

This section provides a concise overview of the contributions of this thesis. Items from 1 to 8 of the following list refer to contributions in the field of Controlled Query Evaluation, whereas items from 9 to 11 are related to Consistent Query Answering.

1. We systematically review the CQE framework for Description Logics, providing formal foundational definitions and highlighting the fundamental characteristics distinguishing the various types of censors. We give a particular focus on the crucial property of indistinguishability, explaining in which cases it is enjoyed.

2. We introduce the notion of IGA censor, which allows to soundly approximate skeptical reasoning by intersecting the outputs of all the optimal GA censors. We show that such a function enjoys the indistinguishability property and, for DL-Lite$_\mathcal{R}$ ontologies, we prove that answering UCQs under the IGA censor is FO rewritable.

3. We introduce several semantics for query answering exploiting a given priority relationship between ontology predicates. With respect to the previous case, these approaches reduce the necessity of lying to the end user while maintaining confidentiality. We identify a well-founded semantics extending the IGA one for which UCQ answering is still tractable, and a parameterized version thereof for which we provide a suitable FO-rewriting technique for DL-Lite$_\mathcal{R}$ ontologies.

4. We show how to extend the aforementioned results to the case in which the ontology is expressed in DL-Lite$_\mathcal{A}$, a logic enhancing DL-Lite$_\mathcal{R}$ with the possibility of using *data values* (like strings, integers, etc.), and the policy language allows for specifying comparisons between such data values (e.g. $\mathsf{age} < 18$) in a controlled way.

5. We also show how to extend our results to the case in which the ontology is represented in DL-Lite$^{\mathcal{H}}_{horn}$ and the policy may include *numerical restriction axioms*, i.e. logical formulas expressing conditions based on the number of distinct relationships an individual has with other individuals.

6. We consider the *policy-protected ontology-based data access* (*PPOBDA*) case, i.e. when OBDA specification is coupled with a data protection policy, and we show how to encode the problem of privacy-preserving UCQ answering in such a setting into a standard OBDA query answering task. We prove that the above problem has in general no solution when either all CQ censors or all GA censors are considered. We then provide an algorithm that realizes such an encoding for the aforementioned FO rewritable semantics for prioritized CQE in case of DL-Lite$_{\mathcal{A}}$ ontologies and policies featuring comparison atoms.

7. We show the practical applicability of our approach through an experimental study over the NPD benchmark [79]. Specifically, we developed a software that, starting from a PPOBDA specification, generates a new OBDA mapping accounting for all the rules specified in the policy. The resulting mapping has subsequently been used, following the intuition described earlier, for testing the scenario mentioned in the previous point. Our experiments show that prioritized CQE is feasible in practice and that priorities are particularly effective in increasing the amount of data disclosed to the user, still guaranteeing confidentiality preservation.

8. We formalize the case in which a stream of queries is answered dynamically, defining a CQE semantics that enjoys the longest-honeymoon property. We also prove that any maximally cooperative semantics for answering (Boolean) UCQs coincides with the one that we defined. Finally, we provide an FO rewriting technique for solving the related entailment problem in the case of DL-Lite$_{\mathcal{R}}$ ontologies.

9. We study the repair checking problem and provide data complexity results for all possible combinations of the considered subclasses of dependencies. Among the tractable settings, we present two FO-rewriting techniques for solving the case in which dependencies are acyclic and the case in which they are either full+sticky or full+linear.

10. We study IAR- and AR-entailment for DED$^{\neq}$s and provide data complexity results for all possible combinations of the considered subclasses of dependencies. Among the tractable settings, we present two FO-rewriting techniques for solving IAR-entailment when dependencies are acyclic+linear and the case in which they are acyclic+full, full+sticky, or full+linear. Notably, the first and the last cases are also valid for the AR semantics (as, when dependencies are linear, the two entailment problems coincide). Moreover, all the lower bounds proved for IAR-entailment already hold when the query posed consists of a single ground atom.

11. We extend the definition of repair to handle open and closed predicates, and we introduce the notion of OCS dependencies. We finally provide data complexity

results for repair checking, IAR- and AR-entailment for several classes of OCS dependencies.

1.2 Thesis structure overview

The thesis is structured as outlined below.

- Chapter 2 serves as a foundational component of the thesis, providing a comprehensive presentation of all technical concepts and background information essential for readers to grasp the subsequent content effectively.
- In Chapter 3, we formally describe the CQE framework, the notion of censors, and significant formal properties thereof. We also provide some preliminary results already known in the literature. Finally, we introduce and analyze the case of the IGA censor.
- Chapter 4 focuses on prioritized CQE, explaining how to exploit a priority relation between ontology predicates for improving the throughput of query answers.
- In Chapter 5, we extend the CQE framework in two directions, both of which involve enhancing the ontology language and policy language.
- In Chapter 6, we discuss how to encode CQE in the problem of answering queries over an OBDA system. We then illustrate the experiments we performed in the OBDA scenario.
- In Chapter 7, we present a further extension of CQE, i.e. the dynamic case, and we show a practical technique agreeing with the "longest honeymoon" approach.
- In Chapter 8, we formally describe the CQA framework and the semantics considered for query answering. We then present several complexity results for the studied decision problems.
- In Chapter 9, we discuss how to integrate closed- and open-world assumptions in the context of CQA and we present preliminary complexity results for repair checking and inconsistency-tolerant query answering with OCS dependencies.
- Chapter 10 offers an extensive review of prior studies' methodologies and findings in the field, providing valuable context for the original contributions and insights presented in the thesis.
- In Chapter 11, we conclude the thesis by providing a succinct discussion of the presented results and exploring prospects for future research directions.

Chapter 2

Preliminaries

2.1 First-order logic

In this thesis, we refer to the classical notions of first-order (FO) logic without function symbols (except for constants). Any FO language comes with three countably infinite, pairwise disjoint sets, namely $\mathscr{P}$ for *predicates*, $\mathscr{C}$ for *constants* and $\mathscr{V}$ for *variables*. Each predicate $P \in \mathscr{P}$ is given with an associated non-negative integer called *arity*, denoting how many arguments P takes as input. Constants and variables are more generally called *terms*.

Syntax. The syntax for function-free FO logic is the usual one. A *predicate atom* is an FO expression of the form $P(\mathbf{t})$, where $P \in \mathscr{P}$ is a predicate of arity n and $\mathbf{t}$ an n-ary sequence of terms[1]. Any FO expression is said to be *ground* if it does not contain any variable. Predicate atoms that are also ground are commonly simply called *ground atoms* or *facts*. A *comparison atom* is an expression of the form $t_1 \,\mathrm{op}\, t_2$, where $\mathrm{op} \in \{=, \neq, <, \leq, \geq, >\}$ and t_1 and t_2 are terms. In particular, we refer to the forms $t_1 = t_2$ and $t_1 \neq t_2$ as *equality atoms* (or simply *equalities*) and *inequality atoms* (or simply *inequalities*), respectively. Predicate and comparison atoms are the building blocks of FO formulas, whose form is the usual one and may include logical connectives as $\neg$, $\wedge$, $\vee$, $\rightarrow$,[2] universal and existential quantifiers (namely, $\forall$ and $\exists$), and parenthesis, when needed for clarifying precedence between subformulas. Given an FO formula (or a set thereof) ϕ, we denote as $\mathit{pred}(\phi)$ and $\mathit{vars}(\phi)$, respectively, the set of predicates and the set of variables occurring in ϕ. Moreover, we call *predicate signature* (or simply *signature*) of ϕ the set $\mathit{pred}(\phi)$.

Every variable of an FO formula that is neither existentially nor universally quantified is said to be *free*. Given an FO formula ϕ, we also use the notation $\phi(\mathbf{x})$ to emphasize its free variables $\mathbf{x}$, and we write $\mathit{vars}(\phi)$ for indicating the set of all the variables occurring in ϕ. A *closed formula* (also known as *sentence* or *Boolean formula*) is a formula with no free variables occurring in it. Given an FO formula $\phi(\mathbf{x})$ with free variables $\mathbf{x} = x_1, \ldots, x_n$ and a sequence of constants $\mathbf{c} = c_1, \ldots, c_n$, we indicate with $\phi(\mathbf{c})$ the sentence in which every occurrence of the variable x_i has been replaced with c_i, for every i such that $1 \leq i \leq n$. A set of FO sentences is

[1] We use bold letters for indicating sequences of terms, constants or variables.

[2] We assume that the precedence of logical connectives follows the order in which they have been introduced. For example, we write $\alpha \wedge \beta \rightarrow \gamma$ giving precedence to $\wedge$.

called *FO theory*.

Semantics. The semantics of an FO theory is provided through the interpretation of its non-logical symbols (namely, predicates and constants). Formally, an *FO interpretation* (or simply *interpretation*) $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where:

- $\Delta^{\mathcal{I}}$ is a non-empty, possibly-infinite set called *domain of the interpretation*;
- $\cdot^{\mathcal{I}}$ is the *interpretation function*, i.e. a function such that:
  - $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each constant $c \in \mathscr{C}$;
  - $P^{\mathcal{I}} \in (\Delta^{\mathcal{I}})^k$ for each predicate $P \in \mathscr{P}$ of arity k.

We also adopt of the so-called *unique name assumption* (UNA), i.e. we do not allow two distinct constant symbols to be interpreted with the same object of the domain. More formally, for every interpretation $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ for a given FO theory, we have that $c_1^{\mathcal{I}} \neq c_2^{\mathcal{I}}$ for each distinct pair of constants $c_1, c_2 \in \mathscr{C}$. This is a common assumption both in database theory [102, 1] and in the context of Description Logics [10].

Given an interpretation $\mathcal{I}$ and a sentence ϕ, we say that $\mathcal{I}$ *satisfies* ϕ if the standard evaluation of ϕ over $\mathcal{I}$ returns *true*. Given an FO theory $\mathfrak{T}$, every interpretation $\mathcal{I}$ for $\mathfrak{T}$ satisfying all the sentences specified in it is called *model* of $\mathfrak{T}$. If at least one such model exists, we say that $\mathfrak{T}$ is *consistent*, *inconsistent* otherwise. We say that a sentence ϕ is *entailed* by an FO theory $\mathfrak{T}$ (in symbols $\mathfrak{T} \models \phi$) if ϕ is satisfied by every model $\mathcal{I}$ of $\mathfrak{T}$. If this is not the case, then $\mathfrak{T}$ does not entail ϕ (in symbols $\mathfrak{T} \not\models \phi$). Two FO theories $\mathfrak{T}_1$ and $\mathfrak{T}_2$ are *equivalent* if every interpretation is a model of $\mathfrak{T}_1$ if and only if it is a model of $\mathfrak{T}_2$.

Substitutions. We use the term *substitution* [38] for denoting a mapping from variables to terms. A substitution σ takes the form of $\{v_1 \mapsto t_1, \ldots, v_k \mapsto t_k\}$ and maps k distinct variables $v_1, \ldots, v_k \in \mathscr{V}$ to k corresponding terms $t_1, \ldots, t_k \in \mathscr{V} \cup \mathscr{C}$. If every term t_i (with $1 \leq i \leq k$) is a constant, then σ is called a *ground substitution*. Given a formula ϕ, we use the notation $\sigma(\phi)$ for indicating the result of *applying* σ *to* ϕ, i.e. replacing each v_i with the corresponding term t_i, with $1 \leq i \leq k$. For instance, given a substitution $\sigma = \{x \mapsto c, y \mapsto z\}$ and an FO formula $\phi = R(x, y, z)$, we have that σ replaces variables x and y with, respectively, a constant c and a variable z, i.e. $\sigma(\phi) = R(c, z, z)$. We also define the *composition* operator ($\circ$) as follows: given two substitutions σ and σ' and an FO formula ψ, $[\sigma \circ \sigma'](\phi) = \sigma'(\sigma(\phi))$. We remark that composition of substitutions is not commutative.

Given two predicate atoms α and β, a *unifier* for α and β is any substitution σ such that $\sigma(\alpha) = \sigma(\beta)$. A unifier σ is called *most general unifier* (*MGU*) for α and β if, for each unifier σ' for α and β, there exists a substitution σ'' such that $\sigma' = \sigma \circ \sigma''$. If a unifier for α and β exists, then we say that such atoms are *unifiable*. It is well known [38] that (*i*) if α and β are unifiable, then there also exists an MGU for α and β, and (*ii*) if an MGU for α and β exists, then it is unique up to renaming of variables. We finally extend the notions of unifier and MGU to sets of atoms in the obvious way, and we denote as $MGU(\mathcal{F})$ the function that, given a set of unifiable atoms $\mathcal{F} = \{\alpha_1, \ldots, \alpha_k\}$, returns a common MGU for all such atoms.

2.2 Notes on complexity theory

We investigate the complexity of various problems, in fact *decision problems*, which yield a binary answer (either *true* or $false$) for any given input. We say that an algorithm *solves* a decision problem if, for every valid input, its execution is guaranteed to terminate and it returns a correct answer.

Decision problems are typically categorized in different complexity classes, depending on the amount of computational resources that may be required, in the worst case, for producing a correct solution. In this thesis, we primarily focus on the following complexity classes:

- AC^0, defined as the class of decision problems solvable by circuit families of constant (i.e. independent of the input size) depth, polynomial size, and whose gates have unbounded fan-in (each gate can handle an arbitrarily-large number of inputs);
- PTime (or P), defined as the class of decision problems solvable in polynomial time by a deterministic Turing machine;
- NP, defined as the class of decision problems solvable in polynomial time by a non-deterministic Turing machine;
- coNP, defined as the class of decision problems whose complement is solvable in polynomial time by a non-deterministic Turing machine;
- Π_2^p, defined as the class of decision problems whose complement is solvable in polynomial time by a non-deterministic Turing machine augmented by an oracle for some NP-complete problem.

It is well known that that $\mathrm{AC}^0 \subset \mathrm{PTime} \subseteq \mathrm{NP} \cup \mathrm{coNP} \subseteq \Pi_2^p$, while it is not known whether such $\subseteq$-containments are strict or whether $\mathrm{NP} = \mathrm{coNP}$.

Given a complexity class C and a decision problem p, we say that p is *hard* for C (or *C-hard*) if there exists a logspace reduction from every other problem within C to p. If p does also belong to C, we say that p is *C-complete*. We refer the reader to Appendix B for a brief overview of the decision problems employed for reductions.

We finally point out that our analysis focuses on the so-called *data complexity* (a notion borrowed from the database literature [116]), i.e. when examining the complexity of a problem we consider the input data may vary, while all the other inputs are fixed. This complexity metric is especially pertinent in scenarios where the size of the intensional level of the knowledge base is negligible in comparison to the size of the data, as in ontology-based data access [100].

2.3 Description logics

In the first part of this thesis we mainly focus on Description Logics (DLs), [10, 11] that is a class of languages most of which corresponding to fragments of FO logic exclusively using unary and binary predicates. While FO logic offers a powerful means of expressing relationships between objects, its computational complexity

can be limiting for large-scale knowledge bases. DLs address this by offering a balance between expressiveness and computational tractability.

Given a DL denoted as $\mathcal{L}_\mathcal{T}$, an $\mathcal{L}_\mathcal{T}$ *TBox* ("Terminological Box") $\mathcal{T}$ is a finite collection of axioms whose form may vary according to the logic $\mathcal{L}_\mathcal{T}$. Despite the different syntax from standard FO logic, we remark that each TBox axiom can be equivalently expressed as an FO sentence (meaning that a TBox is an FO theory). Unless otherwise indicated, we assume that every TBox is specified over three pairwise disjoint subsets of $\mathscr{P}$, namely Γ_C for *concepts* (a.k.a. *atomic concepts*), Γ_R for *roles* (a.k.a. *atomic roles*) and Γ_A for *attributes*. While concepts are unary predicates, roles and attributes are binary. We also partition the set of constants $\mathscr{C}$ in two disjoint subsets: Γ_I for *individuals* and Γ_V for *values*.

Given a TBox $\mathcal{T}$, an *ABox* ("Assertional Box") $\mathcal{A}$ for $\mathcal{T}$ is a finite set of ground atoms having the form $A(o)$, $P(o, o')$ or $U(o, v)$, where A, P and U are, respectively, a concept, a role and an attribute belonging to the signature of $\mathcal{T}$, o and o' are individuals from Γ_I, and v is a value from Γ_V. Hereinafter, if a TBox $\mathcal{T}$ is given, when we reference an ABox $\mathcal{A}$ it is understood that $\mathcal{A}$ is for $\mathcal{T}$.

In simple words, we can say that the purpose of TBox and ABox is to specify, respectively, the intensional and extensional knowledge of the domain of interest. A broader construct taking both of these aspects into account is typically called an ontology. Formally, given an $\mathcal{L}_\mathcal{T}$ TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, we call $\mathcal{L}_\mathcal{T}$ *ontology* the FO theory $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$. As such, ontologies inherit the model-theoretic semantics of FO theories [10]: the interpretation of a concept, a role or an attribute result, respectively, in a set of individuals, a set of pairs of individuals and a set of pairs of one individual and one value constant, and a *model of* $\mathcal{O}$ is an interpretation satisfying all the assertions in $\mathcal{T}$ and $\mathcal{A}$, where the definition of satisfaction specializes according to the DL $\mathcal{L}_\mathcal{T}$. Unless otherwise indicated, we assume that an ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ is a consistent FO theory.

2.3.1 Syntax of a DL-Lite$_\mathcal{R}$ ontology

In this work, we largely focus on DL-Lite$_\mathcal{R}$ ontologies. DL-Lite$_\mathcal{R}$ is a language belonging to the DL-Lite family [47, 43, 45] of DLs. It is well-known for underpinning OWL 2 QL, one of the three OWL 2 profiles [56, 97] that is specifically designed for efficient query answering. Note that OWL 2 is the standard ontology language recommended by W3C for Semantic Web applications.

DL-Lite$_\mathcal{R}$ concepts and roles are defined according to the following syntax:

$$B ::= A \mid \exists R \qquad R ::= P \mid P^-$$

where A is an atomic concept, B is a *basic concept*, P is an atomic role, and R is a *basic role*. The symbols A, P, B, and R, possibly with subscripts, are used in the following with the above meaning. The expressions $\exists P$ and $\exists P^-$ are known as *unqualified existential restrictions* and denote, respectively, the domain and range of the role P. Moreover, we call P^- the *inverse* of P. Then, a DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$

is a finite set of assertions of the following form:[3]

$B_1 \sqsubseteq B_2$ (positive concept inclusion)
$R_1 \sqsubseteq R_2$ (positive role inclusion)
$B_1 \sqsubseteq \neg B_2$ (negative concept inclusion, or concept disjointness)
$R_1 \sqsubseteq \neg R_2$ (negative role inclusion, or role disjointness)

where B_1 and B_2 are basic concepts and R_1 and R_2 are basic roles.

Let C and S (both possibly subscripted) be, respectively, a basic concept or its negation and a basic role or its negation. In principle, one could also introduce the operators of union $\sqcup$ (resp., intersection $\sqcap$) as follows: $B_1 \sqcup B_2 \sqsubseteq C$, $R_1 \sqcup R_2 \sqsubseteq S$ (resp., $B \sqsubseteq C_1 \sqcap C_2$, $R \sqsubseteq S_1 \sqcap S_2$). Anyway, it is easy to see that the addition of such constructs would not increase the language's expressive power, since they would be equivalent to $B_1 \sqsubseteq C, B_2 \sqsubseteq C$ and $R_1 \sqsubseteq S, R_2 \sqsubseteq S$ (resp., $B \sqsubseteq C_1, B \sqsubseteq C_2$ and $R \sqsubseteq S_1, R \sqsubseteq S_2$). The introduced constructs are also sufficient for expressing other assertions included in OWL 2 QL, namely *qualified existential restriction*, *symmetry*, *asymmetry*. Qualified existential restriction of the form $B_1 \sqcup \exists R.B_2$ can be obtained by exploiting an auxiliary predicate P_{aux} and asserting $B_1 \sqsubseteq \exists P_{\text{aux}}$, $P_{\text{aux}} \sqsubseteq R$ and $\exists P_{\text{aux}}^- \sqsubseteq B_2$. Finally, applying minor changes in reasoning algorithms, one can also express *reflexivity* and *irreflexivity* without affecting the computational complexity of basic reasoning problems [9, 42].

2.3.2 Semantics of a DL-Lite$_\mathcal{R}$ ontology

The semantics of a DL-Lite$_\mathcal{R}$ ontology is defined by interpreting atomic concepts as sets of individuals and atomic roles as sets of pairs of individuals, and reducing the interpretation of basic concepts and roles to the one of atomic concepts and roles. More formally, an interpretation $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ for an ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ satisfies the following equations:

$$\begin{array}{rl} A^\mathcal{I} \subseteq \Delta^\mathcal{I}, & \text{if } A \in \Gamma_I \\ P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}, & \text{if } P \in \Gamma_R \\ (\exists P)^\mathcal{I} = \{o \mid \exists o'(o, o') \in P^\mathcal{I}\}, & \text{if } P \in \Gamma_R \\ (P^-)^\mathcal{I} = \{(o', o) \mid (o, o') \in P^\mathcal{I}\}, & \text{if } P \in \Gamma_R \end{array}$$

The definition of *satisfaction* of a TBox assertion specializes as follows:

$$\begin{array}{lll} \mathcal{I} \models B_1 \sqsubseteq B_2 & \text{iff} & B_1^\mathcal{I} \subseteq B_2^\mathcal{I} \\ \mathcal{I} \models R_1 \sqsubseteq R_2 & \text{iff} & R_1^\mathcal{I} \subseteq R_2^\mathcal{I} \\ \mathcal{I} \models B_1 \sqsubseteq \neg B_2 & \text{iff} & B_1^\mathcal{I} \cap B_2^\mathcal{I} = \emptyset \\ \mathcal{I} \models R_1 \sqsubseteq \neg R_2 & \text{iff} & R_1^\mathcal{I} \cap R_2^\mathcal{I} = \emptyset \end{array}$$

Finally, for ABox assertions we have that $\mathcal{I} \models A(o)$ if and only if $o^\mathcal{I} \in A^\mathcal{I}$, and $\mathcal{I} \models P(o, o')$ if and only if $(o, o')^\mathcal{I} \in P^\mathcal{I}$, for each $A \in \Gamma_C$, $P \in \Gamma_R$ and $o, o' \in \Gamma_I$.

In Chapter 5, we define two other DLs belonging to the DL-Lite family, namely DL-Lite$_\mathcal{A}$ and DL-Lite$_{horn}^\mathcal{H}$, which extend both the syntax and the semantics of DL-Lite$_\mathcal{R}$.

[3] For DL-Lite$_\mathcal{R}$ assertions we adopt the well-known variable-free DL syntax [10].

2.4 Query answering and FO-rewritability

A fundamental task in KR is *query answering*, i.e. the process of formulating and executing queries against a database or knowledge base to retrieve relevant information. In this context, we also use the term *query* as synonym of a domain-independent FO formula [1, 115]. As query language, we focus on the well-known class of conjunctive queries.

Definition 1. [CQ and BCQ] A *conjunctive query with inequalities* ($CQ^{\neq}$) q is an FO formula of the form $\exists \mathbf{x}\, \gamma$ such that $\mathbf{x} \subseteq vars(\gamma)$ and γ is a finite conjunction of atoms of the form $\alpha_1 \wedge \ldots \wedge \alpha_k \wedge \lambda_1 \wedge \ldots \wedge \lambda_h$, where $k \geq 0$, $h \geq 0$, $k + h \geq 1$, every α_i is a predicate atom and every λ_i is an inequality. We use the notation $predAt(q)$ for denoting the set $\{\alpha_i \mid 1 \leq i \leq k\}$. If every variable of q occurs in at least one predicate atom, we say that q is *safe*. If no inequality atoms occur in q, then we simply call it a *conjunctive query* (CQ). Boolean CQ$^{\neq}$s (resp., Boolean CQs) are indicated as $BCQ^{\neq}s$ (resp., $BCQs$).

Definition 2. [UCQ and BUCQ] A *union of conjunctive queries with inequalities* ($UCQ^{\neq}$) q is an FO formula of the form $\bigvee_{i=1}^{m} q_i$, where $m \geq 1$ and every q_i is a CQ$^{\neq}$. We denote by $CQ(q)$ the set of CQ$^{\neq}$s occurring in q, i.e. $CQ(q) = \{q_1, \ldots, q_m\}$. If all such q_i are safe and share the same free variables, we say that q is *safe*. If no inequality atoms occur in q, then we simply call it a *union of conjunctive queries* (UCQ). Boolean UCQ$^{\neq}$s (resp., Boolean UCQs) are indicated as $BUCQ^{\neq}s$ (resp., $BUCQs$).

Note that UCQs and CQs without inequalities are trivially safe, and that every BUCQ$^{\neq}$ consists in a disjunction of BCQ$^{\neq}$s. Throughout this thesis, we refer to safe UCQ$^{\neq}$s and subclasses thereof, omitting that they are safe.

As anticipated in the introduction, we use FO formulas also for specifying the rules contained in the data protection policy. A *denial assertion* (or simply *denial*) δ is an FO sentence of the form $\forall \mathbf{x}\, (\phi(\mathbf{x}) \rightarrow \bot)$, where $\phi(\mathbf{x})$ is a conjunction of predicate atoms. For convenience, we sometimes express a denial in the form $q \rightarrow \bot$, where q is a BCQ (called *premise* of δ), and we use $predAt(\delta)$ for actually denoting $predAt(q)$.

Denials and BUCQ$^{\neq}$s are both classes of domain-independent FO formulas. Given a set of facts $\mathcal{F}$ and a domain-independent FO sentence ϕ over the same signature, we also say that ϕ *evaluates to true (resp., to false) in* $\mathcal{F}$ if the Herbrand model [86] of $\mathcal{F}$ (i.e. the model isomorphic to $\mathcal{F}$) satisfies ϕ.

Let $\mathcal{F}$ be a set of facts, and let $q = \exists \mathbf{x}\, \gamma$ be a CQ$^{\neq}$. An *instantiation of* q *in* $\mathcal{F}$ is a ground substitution σ of the variables occurring in q with constants occurring $\mathcal{F}$ such that

(i) $predAt(\sigma(\gamma)) \subseteq \mathcal{F}$;

(ii) no inequality of the form $c \neq c$, for any $c \in \mathscr{C}$, occurs in $\sigma(\gamma)$.

An *image of* q *in* $\mathcal{F}$ is a subset $\{\sigma(\alpha) \mid \alpha \in predAt(q)\}$ of $\mathcal{F}$, where σ is any instantiation of q in $\mathcal{F}$. Given a UCQ$^{\neq}$ q, an *image of* q *in* $\mathcal{F}$ is any image of q' in $\mathcal{F}$, for some $q' \in CQ(q)$. Given a BUCQ$^{\neq}$ q and a set of facts $\mathcal{F}$, it is immediate to

verify that q evaluates to true in every image of q itself in $\mathcal{F}$, and that q evaluates to true in $\mathcal{F}$ if and only if there exists an image of q in $\mathcal{F}$.

We say that an FO theory $\mathfrak{T}$ *satisfies* a denial δ of the form $q \rightarrow \bot$ if $\mathfrak{T} \not\models q$, otherwise we say that $\mathfrak{T}$ *violates* δ. Note that, if $\mathfrak{T}$ is consistent, then the theory $\mathfrak{T} \cup \{\delta\}$ is consistent if and only if $\mathfrak{T}$ satisfies δ.

Hereinafter, with **FO**, **CQ**, and **GA** we denote, respectively, the languages of function-free FO sentences, BCQs, and ground atoms, all specified over the alphabets $\mathscr{P}, \mathscr{C}$, and $\mathscr{V}$. Note that $\mathbf{GA} \subseteq \mathbf{CQ} \subseteq \mathbf{FO}$. Given an ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ and a language $\mathcal{L} \subseteq \mathbf{FO}$, the notation $\mathcal{L}(\mathcal{O})$ refers to the subset of $\mathcal{L}$ containing all those sentences that can be built using the atomic concepts and atomic roles in the signature of $\mathcal{T}$ as predicates, and the constants occurring in $\mathcal{T} \cup \mathcal{A}$ and the variables in $\mathscr{V}$ as terms. Given a language $\mathcal{L} \subseteq \mathbf{FO}$, a TBox $\mathcal{T}$, and an ABox $\mathcal{A}$, we denote by $cl_{\mathcal{L}}^{\mathcal{T}}(\mathcal{A})$ the set of those sentences in $\mathcal{L}(\mathcal{T} \cup \mathcal{A})$ that are entailed by $\mathcal{T} \cup \mathcal{A}$, i.e. $cl_{\mathcal{L}}^{\mathcal{T}}(\mathcal{A}) = \{\phi \mid \phi \in \mathcal{L}(\mathcal{T} \cup \mathcal{A}) \text{ and } \mathcal{T} \cup \mathcal{A} \models \phi\}$.

We recall that, for every DL-Lite$_{\mathcal{R}}$ TBox $\mathcal{T}$ and BUCQ q, it is possible to effectively compute an FO query q_r, called the *perfect reformulation of q with respect to $\mathcal{T}$*, such that, for each ABox $\mathcal{A}$, we have $\mathcal{T} \cup \mathcal{A} \models q$ if and only if q_r evaluates to true in $\mathcal{A}$ [45]. This yields the well-known result that answering BUCQs over DL-Lite$_{\mathcal{R}}$ ontologies is *FO rewritable*, and then the underlying decision problem is in AC^0 in the size of the ABox (i.e. in data complexity).

In this thesis, we use the algorithm *PerfectRef* given in [45], which takes as input a BUCQ q and a TBox $\mathcal{T}$ and uses the positive inclusions in $\mathcal{T}$ as rewriting rules to compute the perfect reformulation of q with respect to $\mathcal{T}$. *PerfectRef* executes two main steps, applied repeatedly to each BCQ $q' \in CQ(q)$ until a fixpoint is reached: step (i) uses positive inclusions as rewriting rules applied from right to left, to rewrite query atoms one by one, each time producing a new BCQ to be added to the final rewriting q_r; step (ii) unifies the atoms in the query to enable further executions of step (i). A more detailed description of this algorithm is provided in Appendix A. We point out that the reformulation q_r returned by *PerfectRef* is a BUCQ. The main property enjoyed by the algorithm is the following one.

Proposition 1. *Let $\mathcal{T}$ be a DL-Lite$_{\mathcal{R}}$ TBox, and let q be a BUCQ. For every ABox $\mathcal{A}$, we have that $\mathcal{T} \cup \mathcal{A} \models q$ if and only if PerfectRef$(q, \mathcal{T})$ evaluates to true in $\mathcal{A}$.*

For the sake of presentation, we limit our technical treatment to query languages containing only closed formulas. The evaluation problem of (open) UCQs over a database can be characterized as follows: given a set of facts $\mathcal{F}$, a query q, and a ground substitution σ of its free variables, check whether $\sigma(q)$ evaluates to true in $\mathcal{D}$. We call *answer set* of q in $\mathcal{F}$ the set of all such σ.

2.5 Ontology-Based Data Access

As outlined in the introduction, *Ontology-Based Data Access* (*OBDA*) [100, 120, 85] is a framework in which one or more data sources, not necessarily designed with the aim of describing the ontology domain, are enriched with the knowledge provided by a DL TBox. An ABox is not provided as a separated object, but instances of the ontology predicates are represented as virtual views on the data.

A key aspect that needs to be addressed is the disparity in formalisms used to reference ontologies and data sources. Specifically, ontologies are commonly defined using logical languages, whereas data is typically expressed by means of a relational data model. To bridge this gap, the OBDA architecture provides a third layer, namely the mapping layer, which is used to properly link the data with the conceptual view.

The primary and fundamental function within this framework is query answering. When accessing to such a service, the user just need to know how the information is conceptualized, i.e. from her perspective the query is posed to an ontology. The system must then reason with respect to the assertions provided in the ontology and take into account how the mapping is defined, and subsequently convert the client's request into suitable queries that are evaluated on the data sources.

A further crucial concern is choosing an ontology language allowing to model adequately complex domains and, at the same time, offering strong computational properties for reasoning over the entire system. In pursuit of this goal, a good compromise turned out to be DL-Lite$_{\mathcal{R}}$, which, as remarked above, is designed for answering queries efficiently, even with a large amount of underlying data.

We now define the OBDA framework in a formal way. The relational database schema is modeled with a set $\mathcal{S}$, that we call *source schema*, containing relational predicates. We assume that each source schema is specified over an alphabet that is disjoint from $\Gamma_C \cup \Gamma_R \cup \Gamma_A$. Given a source schema $\mathcal{S}$, a *database instance* (or simply *database*) $\mathcal{D}$ for $\mathcal{S}$ is a finite set of ground atoms over $\mathcal{S}$.

A mapping between a source schema $\mathcal{S}$ and a TBox $\mathcal{T}$ is defined as a finite set $\mathcal{M}$ of *mapping assertions* from $\mathcal{S}$ to $\mathcal{T}$. Each assertion $m \in \mathcal{M}$ has the form $\phi(\mathbf{x}) \rightsquigarrow \psi(\mathbf{x})$, where $\phi(\mathbf{x})$ and $\psi(\mathbf{x})$, called respectively the *body* and the *head of* m, are FO queries over the predicates of $\mathcal{S}$ and *pred*($\mathcal{T}$), respectively, both with free variables $\mathbf{x}$. In our treatment, we consider *global-as-view (GAV) mapping assertions* [84, 64], i.e. we assume that $\psi(\mathbf{x})$ is a single-atom query without constants and existential variables. This is the form of mapping commonly adopted in OBDA, and is compliant with the W3C standard R2RML [60]. The intuition behind this kind of mappings is that the extension of each predicate of the ontology is defined in terms of views on a relational database.

In the aforementioned definitions, for the sake of simplicity, we have made the assumption that the identifiers (e.g. the URIs) of the instances of the ontology predicates are directly stored in the data layer. Eliminating this assumption would imply introducing a in the head of mapping assertions a mechanism for generating objects from the database values, as shown in [100]. Such an addition would unnecessarily complicate our technical treatment. Nevertheless, it is worth noting that all our results remain valid even in the absence of this simplification.

Given a DL TBox $\mathcal{T}$, a source schema $\mathcal{S}$ and a mapping $\mathcal{M}$ between $\mathcal{S}$ and $\mathcal{T}$, we call *OBDA specification* the triple $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$. The semantics of $\mathcal{J}$ is given with respect to a database instance for $\mathcal{S}$, called source database for $\mathcal{J}$. Given one such database $\mathcal{D}$, we call *retrieved ABox* for $\mathcal{J}$ with respect to $\mathcal{D}$, denoted as *ret*($\mathcal{J}, \mathcal{D}$), the ABox consisting of all and only the facts $\psi(\mathbf{t})$ such that $\mathcal{M}$ contains a mapping assertion of the form $\phi(\mathbf{x}) \rightsquigarrow \psi(\mathbf{x})$, and $\mathbf{t}$ is a tuple of constants such that $\phi(\mathbf{t})$ evaluates to true in $\mathcal{D}$. In other terms, the retrieved ABox is obtained by evaluating the mapping assertions over the source database. Then, a *model* of $\mathcal{J}$

with respect to $\mathcal{D}$ is any model of the ontology $\mathcal{T} \cup \mathit{ret}(\mathcal{J}, \mathcal{D})$. We refer to $\langle \mathcal{J}, \mathcal{D} \rangle$ as an *OBDA setting*. Given an FO sentence ϕ, we denote by $\langle \mathcal{J}, \mathcal{D} \rangle \models \phi$ if $\langle \mathcal{J}, \mathcal{D} \rangle$ *entails* ϕ, i.e. if ϕ is satisfied by every model $\mathcal{I}$ of $\mathcal{J}$ with respect to $\mathcal{D}$.

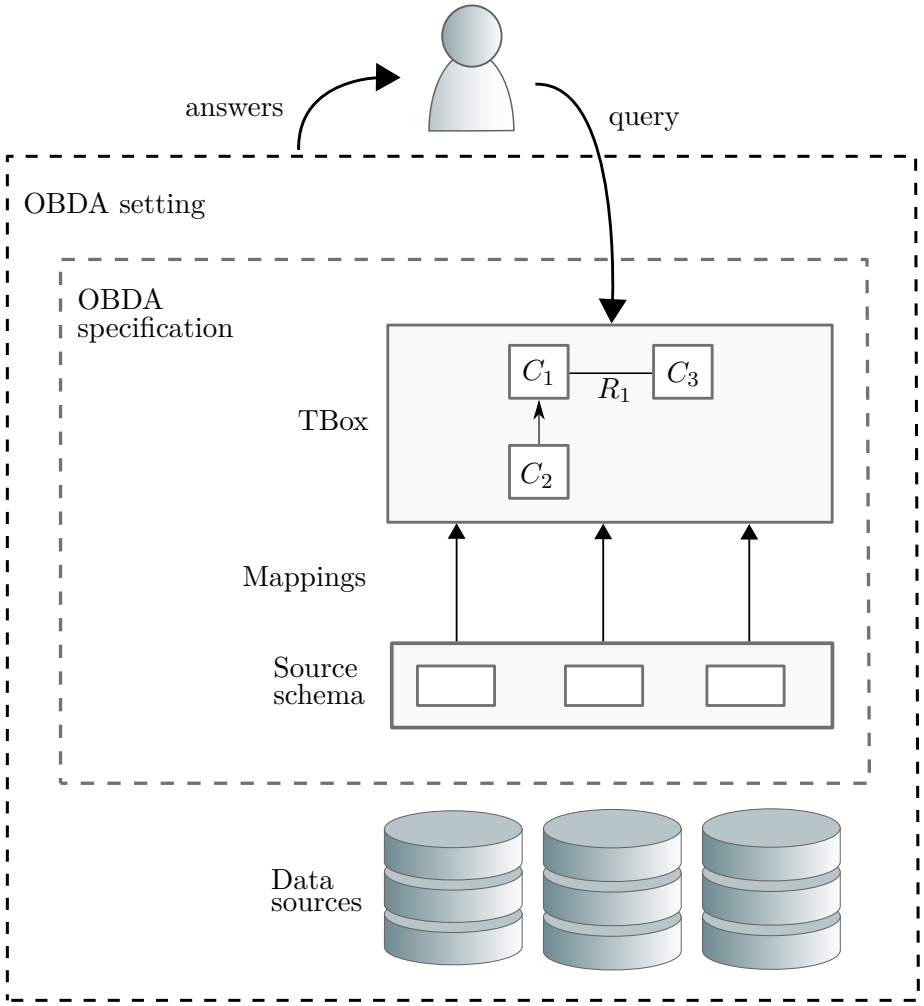

Figure 2.1. Schematization of query answering in OBDA.

Chapter 3

Controlled query evaluation for DL ontologies

3.1 Framework

In this section, we introduce the Controlled Query Evaluation (CQE) framework with a particular focus on DLs. As anticipated in the introduction, we consider a data protection policy that is declaratively specified trough a set of rules that must be enforced by the system. Formally, given a TBox $\mathcal{T}$, we define a *policy* $\mathcal{P}$ for $\mathcal{T}$ as a finite set of denials over the signature of $\mathcal{T}$ such that $\mathcal{T} \cup \mathcal{P}$ is a consistent FO theory. We note that, given any ABox $\mathcal{A}$, the FO theory $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}$ is consistent iff, for each $\delta \in \mathcal{P}$, the FO theory $\mathcal{T} \cup \{\delta\} \cup \mathcal{A}$ is consistent.

We can now introduce the notion of CQE specification.

Definition 3. [CQE specification] Let $\mathcal{L}_\mathcal{T}$ be a DL, let $\mathcal{T}$ be an $\mathcal{L}_\mathcal{T}$ TBox and let $\mathcal{P}$ be a policy for $\mathcal{T}$. We call the pair $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ an $\mathcal{L}_\mathcal{T}$ *CQE specification*.

Hereinafter, when a TBox $\mathcal{T}$ is given we always assume that any coupled policy $\mathcal{P}$ is for $\mathcal{T}$. Moreover, we do not specify $\mathcal{L}_\mathcal{T}$ when a definition or a result applies to any DL.

Example 4. Consider the DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, where:

$$\begin{aligned}
\mathcal{T} \ = \{\ & \mathsf{SServ} \sqsubseteq \mathsf{GAgency}, \\
& \mathsf{SServ} \sqsubseteq \exists \mathsf{OperatesIn}, \\
& \mathsf{Manages} \sqsubseteq \mathsf{WorksFor}, \\
& \exists \mathsf{BreachedBy}^- \sqsubseteq \exists \mathsf{OperatesIn}\ \} \\
\mathcal{P} \ = \{\ & \forall x, y\, (\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y, x) \rightarrow \bot), \\
& \forall x, y\, (\mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(x, y) \rightarrow \bot)\ \}
\end{aligned}$$

In words, the TBox $\mathcal{T}$ sanctions that:

(i) every secret service (SServ) is a particular kind of government agency ($\mathsf{GAgency}$) and it must operate in ($\exists\mathsf{OperatesIn}$) some countries;

(*ii*) if somebody manages (Manages) an agency, then she/he works for (WorksFor) that agency;
(*iii*) who breaches an agency ($\exists$BreachedBy$^-$) operates in ($\exists$OperatesIn) some countries.

The data protection policy specified by $\mathcal{P}$ is intended to hide all the people working for a secret service and all the secret services that have been breached. ▽

We shall now define the notion of censor, which was introduced in [57] as a function yielding a subset of all the sentences that can be inferred from the ontology. In that work, a censor was called *confidentiality-preserving* if, for every dataset given as input, the information provided by the FO theory that it returns was not sufficient for violating the policy. The significance of this last property in effectively enforcing a data-protection policy is self-evident, which is why we encode it into the forthcoming definition of censor outlined below. Such a definition, sourced from [83], also provides a parameterization with respect to a so-called censor language, i.e. the FO language which the sentences returned by the censor belong to.

Definition 4. [Censor] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, and let $\mathcal{L}_c \subseteq \mathbf{FO}$ be a language, called *censor language*. A *censor* in $\mathcal{L}_c$ for $\mathcal{E}$ is a function $cens(\cdot)$ that, for every ABox $\mathcal{A}$, returns a set $cens(\mathcal{A}) \subseteq cl^{\mathcal{T}}_{\mathcal{L}_c}(\mathcal{A})$ such that the FO theory $\mathcal{T} \cup \mathcal{P} \cup cens(\mathcal{A})$ is consistent.

Informally, given a language $\mathcal{L}_c$, a censor in $\mathcal{L}_c$ is a function that, taken an ABox $\mathcal{A}$ for $\mathcal{T}$ as input, returns a set of sentences that (*i*) can be expressed in $\mathcal{L}_c$, (*ii*) can be logically deduced by $\mathcal{T} \cup \mathcal{A}$ and (*iii*) can be safely disclosed to the user without violating the policy $\mathcal{P}$.

Example 5. Consider the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ provided in Example 4, and consider also the following two functions:

cens$_1(\cdot)$: the function such that, given an ABox $\mathcal{A}$, returns the set $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A}) \setminus \{\mathsf{SServ}(c) \mid c \in \Gamma_I \text{ and } \mathcal{T} \cup \mathcal{A} \models \exists x\, (\mathsf{WorksFor}(x, c) \vee \mathsf{BreachedBy}(c, x))\}$;

cens$_2(\cdot)$: the function such that, given an ABox $\mathcal{A}$, returns the set $\mathcal{A} \setminus \{\mathsf{SServ}(c) \mid c \in \Gamma_I \text{ and } \mathsf{SServ}(c) \in \mathcal{A}\}$.

Specifically, the first function returns all the facts that can be concluded from the ontology $\mathcal{T} \cup \mathcal{A}$ except for all the facts $\mathsf{SServ}(c)$ such that, for some other constant c', we have either $\mathsf{WorksFor}(c', c)$ or $\mathsf{BreachedBy}(c, c')$ is contained in the ontology. On the other hand, the second one is returning all the facts of the ABox except for all the instances of SServ. Clearly, we have that both $cens_1(\cdot)$ and $cens_2(\cdot)$ are censors in **GA** for $\mathcal{E}$ (i.e. censors returning sets of facts). ▽

Example 6. Let again $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be as in Example 4. The following functions are censors in **CQ** for $\mathcal{E}$ (i.e. censors returning sets of BCQs).

cens$_3(\cdot)$: the function that, given an ABox $\mathcal{A}$, returns the set computed by removing from $cl^{\mathcal{T}}_{\mathbf{CQ}}(\mathcal{A})$ all the BCQs containing at least one of the following subformulas:

(*i*) a conjunction $\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y, x)$;
(*ii*) a conjunction $\mathsf{SServ}(x) \wedge \mathsf{Manages}(y, x)$;
(*iii*) a conjunction $\mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(x, y)$;
(*iv*) an atom $\mathsf{WorksFor}(x, a)$, with $(a, x) \in \Gamma_I \times \mathscr{V}$, s.t. $\mathcal{T} \cup \mathcal{A} \models \mathsf{SServ}(a)$;
(*v*) an atom $\mathsf{Manages}(x, a)$, with $(a, x) \in \Gamma_I \times \mathscr{V}$, s.t. $\mathcal{T} \cup \mathcal{A} \models \mathsf{SServ}(a)$;
(*vi*) an atom $\mathsf{BreachedBy}(a, x)$, with $(a, x) \in \Gamma_I \times \mathscr{V}$, s.t. $\mathcal{T} \cup \mathcal{A} \models \mathsf{SServ}(a)$.

$\boldsymbol{cens}_4(\cdot)$: the function that, given an ABox $\mathcal{A}$, returns the set computed by removing from $cl^{\mathcal{T}}_{\mathbf{CQ}}(\mathcal{A})$ all the BCQs containing at least one atom between $\mathsf{WorksFor}(t_1, t_2)$ and $\mathsf{BreachedBy}(t_1, t_2)$, where t_1, t_2 are terms in $\Gamma_I \cup \mathscr{V}$. ▽

It is easy to see that a censor always exists[1]. However, as shown by Examples 5 and 6, given a censor language $\mathcal{L}_c$ and a CQE specification $\mathcal{E}$, there may be multiple censors in $\mathcal{L}_c$ for $\mathcal{E}$. Among them, it is reasonable to consider all and only the ones preserving as much information as possible. Formally, given a censor language $\mathcal{L}_c$, a CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ and two censors $cens(\cdot)$ and $cens'(\cdot)$ in $\mathcal{L}_c$ for $\mathcal{E}$, we say that $cens'(\cdot)$ is *more informative* than $cens(\cdot)$ if:

- $cens(\mathcal{A}) \subseteq cens'(\mathcal{A})$, for every ABox $\mathcal{A}$, and
- there exists an ABox $\mathcal{A}'$ such that $cens(\mathcal{A}') \subset cens'(\mathcal{A}')$.

We are now ready to provide a formal definition of optimal censor for a CQE specification.

Definition 5. [Optimal censor] Given a CQE specification $\mathcal{E}$ and a language $\mathcal{L}_c$, we say that a censor $cens(\cdot)$ in $\mathcal{L}_c$ for $\mathcal{E}$ is *optimal* in $\mathcal{L}_c$ if there does not exist any censor $cens'(\cdot)$ in $\mathcal{L}_c$ for $\mathcal{E}$ such that $cens'(\cdot)$ is more informative than $cens(\cdot)$.

Example 7. Consider again the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ and the censors $cens_1(\cdot)$ and $cens_2(\cdot)$ in **GA** for $\mathcal{E}$ of Example 5. Note that $cens_2(\cdot)$ is not optimal in **GA** because $cens_1(\cdot)$ is more informative than $cens_2(\cdot)$. Indeed, by taking the following ABox:

$$\mathcal{A} = \{\ \mathsf{WorksFor}(\mathsf{bob}, \mathsf{a}_1),\ \mathsf{Manages}(\mathsf{ann}, \mathsf{a}_2),\ \mathsf{SServ}(\mathsf{a}_2),\ \mathsf{BreachedBy}(\mathsf{a}_2, \mathsf{h}_2)\ \}$$

it is easy to verify that $cens_2(\mathcal{A}) \subset cens_1(\mathcal{A})$ (in particular, $\mathsf{GAgency}(\mathsf{a}_2) \in cens_1(\mathcal{A}) \setminus cens_2(\mathcal{A})$). Moreover, one can see and that $cens_2(\mathcal{A}') \subseteq cens_1(\mathcal{A}')$ for every other ABox $\mathcal{A}'$. On the other hand, one can verify that $cens_1(\cdot)$ is an optimal censor in **GA** for $\mathcal{E}$. Intuitively, both $cens_2(\cdot)$ and $cens_1(\cdot)$ are exclusively hiding SServ-facts but, unlike the first one, the function $cens_1(\cdot)$ is also censoring facts which could be safely disclosed. ▽

Example 8. Consider again the CQE specification $\mathcal{E}$ of the running example, and recall the censors of Example 6. We have that $cens_3(\cdot)$ is an optimal censor in **CQ** for $\mathcal{E}$ while $cens_4(\cdot)$ is not, since $cens_3(\cdot)$ is more informative than $cens_4(\cdot)$.

[1] As trivial case, note that the function $cens(\cdot)$ such that $cens(\mathcal{A}) = \emptyset$ for each ABox $\mathcal{A}$ is a censor in **GA** for any CQE specification.

Indeed, considering the ABox $\mathcal{A}$ of Example 7, one can verify that $cens_4(\mathcal{A}) \subset cens_3(\mathcal{A})$. Moreover, from the definition of such functions, it is possible to see that $cens_4(\mathcal{A}') \subseteq cens_3(\mathcal{A}')$ for any ABox $\mathcal{A}'$. Intuitively, $cens_4(\cdot)$ hides all the BCQs containing atoms over the roles $\mathsf{WorksFor}$, $\mathsf{Manages}$ and $\mathsf{BreachedBy}$, but some of them could be disclosed without violating the policy, like $\exists x, y\, \mathsf{WorksFor}(x, y)$ or even $\mathsf{WorksFor}(\mathsf{bob}, \mathsf{a_1})$. ▽

For a given censor language $\mathcal{L}_c$, we are naturally interested in the censors in $\mathcal{L}_c$ that are maximally informative, i.e. that do not filter out non-confidential information. Therefore, from now on we restrict our attention to optimal censors.

Moreover, as said in the introduction, to increase the robustness of censors, literature on CQE has often looked at censors satisfying a property of *instance indistinguishability* (or, simply, *indistinguishability*) [32, 37, 19, 52, 35]. Intuitively, a censor fulfilling such a property masks confidential information in such a way that a user cannot distinguish an instance actually containing sensitive data (i.e. those protected by the policy) from an instance without such data, in order to increase the incompleteness of the information of a possible attacker.

A censor for a CQE specification $\mathcal{E}$ satisfies the indistinguishability property if, for every ABox $\mathcal{A}$, there exists an ABox $\mathcal{A}'$ (possibly distinct from $\mathcal{A}$) that does not violate any data protection property and is indistinguishable from $\mathcal{A}$ from the user perspective. In our framework, this is formalized as follows.

Definition 6. [Indistinguishability] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{L}_c \subseteq \mathbf{FO}$, and let $cens(\cdot)$ be a censor in $\mathcal{L}_c$ for $\mathcal{E}$. We say that $cens(\cdot)$ *satisfies the indistinguishability property* if, for every ABox $\mathcal{A}$, there exists an ABox $\mathcal{A}'$ (not necessarily distinct from $\mathcal{A}$) such that:

(*i*) $cens(\mathcal{A}) = cens(\mathcal{A}')$, and

(*ii*) $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}'$ is a consistent FO theory.

Example 9. Consider again the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ and the optimal censor $cens_1(\cdot)$ in **GA** for $\mathcal{E}$ of Example 5. Observe that $cens_1(\cdot)$ satisfies the indistinguishability property because, given any ABox $\mathcal{A}$, we have that $cens_1(\mathcal{A})$ itself plays the role of the ABox $\mathcal{A}'$ satisfying conditions (*i*) and (*ii*) of Definition 6. ▽

Previous research in CQE has mostly considered two censor languages: **CQ** [57, 83] and **GA** [83, 52]. We now analyze these two languages with respect to the above indistinguishability property. For sake of conciseness, hereinafter we refer to any censor in **CQ** (resp., in **GA**) for $\mathcal{E}$ as *CQ censor* (resp., *GA censor*) for $\mathcal{E}$. Analogously, we call *optimal CQ censor* (resp., *optimal GA censor*) the optimal censor in **CQ** (resp., in **GA**), and we denote by $optCQCens(\mathcal{E})$ and $optGACens(\mathcal{E})$ the sets of all optimal CQ censors for $\mathcal{E}$ and all optimal GA for $\mathcal{E}$, respectively.

Unfortunately, as already observed in [52], in general, CQ censors do not enjoy the indistinguishability property. This is illustrated by the following example.

Example 10. Consider the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, where:

$$\mathcal{T} = \emptyset,$$
$$\mathcal{P} = \{\forall x\,(R(x, b) \rightarrow \bot)\},$$

and b is a constant symbol. Note that $optCQCens(\mathcal{E})$ contains only one censor, which is:

cens$(\cdot)$: given an ABox $\mathcal{A}$, $cens(\mathcal{A})$ returns the set computed by removing from $cl^{\mathcal{T}}_{\mathbf{CQ}}(\mathcal{A})$ all the BCQs containing at least one atom over the predicate R having in the second position the constant b.

Consider the ABox $\mathcal{A} = \{R(a, b)\}$. We have that $cens(\mathcal{A})$ is semantically equivalent to $\{\exists x\, R(a, x)\}$. According to Definition 6, there must exists an ABox $\mathcal{A}'$ such that (i) $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}'$ is consistent and (ii) $cens(\mathcal{A}) = cens(\mathcal{A}')$. Since the TBox is empty, the only way to entail the query $\exists x\, R(a, x)$ is to have in $\mathcal{A}'$ an atom of the form $R(a, \alpha)$, where α is a constant. If $\alpha = b$, then $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}'$ is not consistent, while for every $\alpha \neq b$ we have that $R(a, \alpha) \in cens(\mathcal{A}')$ but $R(a, \alpha) \notin cens(\mathcal{A}')$. ▽

We now consider the case of **GA** as the censor language and show that, whatever the CQE specification $\mathcal{E}$, all optimal GA censors for $\mathcal{E}$ enjoy the indistinguishability property. The formal proof makes use of the following two lemmas.

Lemma 1. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $cens(\cdot)$ be an optimal censor in $\mathcal{L}_c$ for $\mathcal{E}$, and let $\mathcal{A}$ be an ABox such that $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}$ is consistent. We have that $cens(\mathcal{A}) = cl^{\mathcal{T}}_{\mathcal{L}_c}(\mathcal{A})$.*

Proof. Note that $cens(\mathcal{A}) \subseteq cl^{\mathcal{T}}_{\mathcal{L}_c}(\mathcal{A})$ trivially holds from the fact that $cens(\cdot)$ is a censor in $\mathcal{L}_c$ for $\mathcal{E}$. We now prove that $cl^{\mathcal{T}}_{\mathcal{L}_c}(\mathcal{A}) \subseteq cens(\mathcal{A})$ holds as well.

Towards a contradiction, suppose this is not the case, i.e. there exists a $\alpha \in cl^{\mathcal{T}}_{\mathcal{L}_c}(\mathcal{A})$ such that $\alpha \notin cens(\mathcal{A})$. Since $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}$ is consistent, and since $\alpha \in cl^{\mathcal{T}}_{\mathcal{L}_c}(\mathcal{A})$, we have $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A} \cup \{\alpha\}$ is consistent as well. So, consider the function $cens'(\cdot)$ such that (i) $cens(\mathcal{A}') = cens'(\mathcal{A}')$ for each ABox $\mathcal{A}' \neq \mathcal{A}$, and (ii) $cens'(\mathcal{A}) = cens(\mathcal{A}) \cup \{\alpha\}$. It is straightforward to verify that $cens'(\cdot)$ is a censor in $\mathcal{L}_c$ for $\mathcal{E}$ and that $cens'(\cdot)$ is more informative than $cens(\cdot)$. This contradicts the fact that $cens(\cdot)$ is an optimal censor in $\mathcal{L}_c$ for $\mathcal{E}$. ∎

In words, the above result shows that, if the considered ABox does not contain confidential data, then every optimal censor in $\mathcal{L}_c$ preserves all the information implied by $\mathcal{A}$ that can be expressed in $\mathcal{L}_c$. In particular, if $\mathcal{L}_c \supseteq \mathcal{A}$, then each optimal censor preserves all the information in $\mathcal{A}$.

Lemma 2. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $cens(\cdot)$ be an optimal censor in $\mathcal{L}_c$ for $\mathcal{E}$, and let $\mathcal{A}$ be an ABox. We have that $cens(\mathcal{A}) = cl^{\mathcal{T}}_{\mathcal{L}_c}(cens(\mathcal{A}))$.*

Proof. Note that $cens(\mathcal{A}) \subseteq cl^{\mathcal{T}}_{\mathcal{L}_c}(cens(\mathcal{A}))$ trivially holds. We now prove that $cl^{\mathcal{T}}_{\mathcal{L}_c}(cens(\mathcal{A})) \subseteq cens(\mathcal{A})$ holds as well.

Towards a contradiction, suppose this is not the case, i.e. there exists a $\alpha \in cl^{\mathcal{T}}_{\mathcal{L}_c}(cens(\mathcal{A}))$ such that $\alpha \notin cens(\mathcal{A})$. Since $\mathcal{T} \cup \mathcal{P} \cup cens(\mathcal{A})$ is consistent because $cens(\cdot)$ is a censor in $\mathcal{L}_c$ for $\mathcal{E}$, and since $\alpha \in cl^{\mathcal{T}}_{\mathcal{L}_c}(cens(\mathcal{A}))$, we have that $\mathcal{T} \cup \mathcal{P} \cup$

$cens(\mathcal{A}) \cup \{\alpha\}$ is consistent as well. So, exactly as done in the proof of Lemma 1, consider the function $cens'(\cdot)$ such that (i) $cens(\mathcal{A}') = cens'(\mathcal{A}')$ for each ABox $\mathcal{A}' \neq \mathcal{A}$, and (ii) $cens'(\mathcal{A}) = cens(\mathcal{A}) \cup \{\alpha\}$. It is straightforward to verify that $cens'(\cdot)$ is a censor in $\mathcal{L}_c$ for $\mathcal{E}$ and that $cens'(\cdot)$ is more informative than $cens(\cdot)$. This contradicts the fact that $cens(\cdot)$ is an optimal censor in $\mathcal{L}_c$ for $\mathcal{E}$. ∎

Theorem 1. *Let $\mathcal{E}$ be a CQE specification, and let $cens(\cdot)$ be an optimal GA censor for $\mathcal{E}$. Then, $cens(\cdot)$ satisfies the indistinguishability property.*

Proof. According to Definition 4, when the censor language is set to **GA**, for every ABox $\mathcal{A}$ we have that $cens(\mathcal{A}) \subseteq cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, i.e. $cens(\mathcal{A})$ is an ABox, and also that $\mathcal{T} \cup \mathcal{P} \cup cens(\mathcal{A})$ is consistent. We show that the ABox $\mathcal{A}'$ of Definition 6 is $cens(\mathcal{A})$ itself, i.e. we show that $cens(\mathcal{A}) = cens(cens(\mathcal{A}))$ for each ABox $\mathcal{A}$.

Consider any ABox $\mathcal{A}$. By Lemma 2, we have that $cens(\mathcal{A}) = cl^{\mathcal{T}}_{\mathbf{GA}}(cens(\mathcal{A}))$. Furthermore, since $\mathcal{T} \cup \mathcal{P} \cup cens(\mathcal{A})$ is consistent, then by Lemma 1 it must hold $cens(cens(\mathcal{A})) = cl^{\mathcal{T}}_{\mathbf{GA}}(cens(\mathcal{A}))$. Thus, $cens(\mathcal{A}) = cl^{\mathcal{T}}_{\mathbf{GA}}(cens(\mathcal{A})) = cens(cens(\mathcal{A}))$, as required. ∎

As a result of the above comparison between CQ and GA censors, since we consider the indistinguishability property a crucial one for safeguarding data confidentiality, in the remaining part of our treatment we focus on optimal GA censors.

3.2 Query answering under optimal GA censors

In this section, we study query answering under optimal GA censors over DL-Lite$_{\mathcal{R}}$ CQE specifications. In particular, we consider entailment of BUCQs.

A possible strategy for addressing this problem is to choose only one GA censor among the optimal ones and use it to alter the answers to user queries. In the absence of a criterion for determining which optimal censor is the best for users' purposes, the choice is made in an arbitrary way, as done in [33, 57].

In the same spirit, in [52] the authors provided the algorithm *OptGACensor* (Algorithm 1), which implements a function that, for every DL-Lite$_{\mathcal{R}}$ CQE specification $\mathcal{E}$ and every ABox $\mathcal{A}$, returns an ABox that coincides with $cens(\mathcal{A})$ for an optimal GA censor $cens(\cdot)$ for $\mathcal{E}$.

The algorithm first computes the set $\mathcal{A}_{\mathcal{T}}$ of ground atoms entailed by $\mathcal{T} \cup \mathcal{A}$. Then, it iteratively picks a ground atom α from $\mathcal{A}_{\mathcal{T}}$ following an arbitrarily-chosen strict total order (e.g. the lexicographic order), and adds α to the current ABox $\mathcal{A}'$ if $\mathcal{T} \cup \mathcal{A}' \cup \{\alpha\}$ does not violate the policy $\mathcal{P}$.

It is easy to see that the algorithm runs in polynomial time in the size of $\mathcal{A}$, as stated by the following theorem, which also establishes the correctness of the algorithm.

Proposition 2 ([52])**.** *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_{\mathcal{R}}$ CQE specification. The following holds:*

- *There exists an optimal GA censor $cens(\cdot)$ for $\mathcal{E}$ such that, for each ABox $\mathcal{A}$, $OptGACensor(\mathcal{E}, \mathcal{A})$ returns $cens(\mathcal{A})$.*

Algorithm 1 *OptGACensor*

input: a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, an ABox $\mathcal{A}$;
output: an ABox $\mathcal{A}'$;
1: **let** $<$ be a strict total order on $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$;
2: $\mathcal{A}_\mathcal{T} \leftarrow cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$;
3: $\mathcal{A}' \leftarrow \emptyset$;
4: **while** $\mathcal{A}_\mathcal{T} \neq \emptyset$ **do**
5: **let** α be the first assertion in $\mathcal{A}_\mathcal{T}$ according to $<$;
6: $\mathcal{A}_\mathcal{T} \leftarrow \mathcal{A}_\mathcal{T} \setminus \{\alpha\}$;
7: **if** $\mathcal{T} \cup \mathcal{A}' \cup \{\alpha\} \cup \mathcal{P}$ is consistent **then**
8: $\mathcal{A}' \leftarrow \mathcal{A}' \cup \{\alpha\}$;
9: **return** $\mathcal{A}'$;

- *Given any ABox $\mathcal{A}$, OptGACensor$(\mathcal{E}, \mathcal{A})$ runs in polynomial time in the size of $\mathcal{A}$;*

From Proposition 2 it follows that, given a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, an ABox $\mathcal{A}$ and a BUCQ q, it is possible to verify in polynomial time in the size of $\mathcal{A}$ whether $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$, where $\mathit{cens}(\cdot)$ is the optimal GA censor for $\mathcal{E}$ implemented by the algorithm (since the size of $\mathit{cens}(\mathcal{A})$ is polinomially related to the size of $\mathcal{A}$ and that deciding $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$ is in AC^0 with respect to the size of $\mathit{cens}(\mathcal{A})$).

Moreover, given a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}$ and an ABox $\mathcal{A}$, it is possible to simulate the behavior of different optimal GA censors for $\mathcal{E}$ (actually, every optimal GA censor for $\mathcal{E}$) on $\mathcal{A}$ by simply modifying the order in which the ABox assertions from the set $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ are selected by the algorithm (see step 4). More formally, it is easy to see that the following holds.

Proposition 3. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{R}$ CQE specification. For every optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ and for every ABox $\mathcal{A}$, there exists a strict total order on the ground atoms in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ for which OptGACensor$(\mathcal{E}, \mathcal{A})$ returns $\mathit{cens}(\mathcal{A})$.*

Depending on the application at hand, however, the approach of randomly choosing a censor may not always be considered appropriate [57]. For this reason, the work [83] studies *skeptical entailment* over all the optimal censors, i.e. a Boolean query q has a positive answer over a CQE specification $\mathcal{E}$ and an ABox $\mathcal{A}$ if q is entailed when considering every optimal GA censor for $\mathcal{E}$.

Definition 7. [GA-Cens-Ent] Given as input a CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, an ABox $\mathcal{A}$, and a Boolean query q, GA-Cens-Ent is the problem of deciding whether $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$ for every optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$. In case this holds, we write $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{GA}} q$.

Example 11. Recall the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ of the running example and the GA censor $\mathit{cens}_1(\cdot)$ for $\mathcal{E}$ illustrated in Example 5. Furthermore, consider the GA censor $\mathit{cens}_4(\cdot)$ for $\mathcal{E}$ such that $\mathit{cens}_4(\mathcal{A}) = cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A}')$ for each ABox $\mathcal{A}$, where $\mathcal{A}'$ is the ABox obtained from $\mathcal{A}$ by removing the ground atoms $\mathsf{Manages}(a, b)$, $\mathsf{WorksFor}(a, b)$, and $\mathsf{BreachedBy}(b, a)$ for all the individuals $a, b \in \Gamma_I$ such that

$\mathsf{SServ}(b) \in \mathcal{A}$. Now, if we consider the ABox $\mathcal{A}$ of Example 7, then it is easy to verify that every optimal GA censor for $\mathcal{E}$ applied to $\mathcal{A}$ returns one of the two following sets of ground atoms:

$$\begin{aligned} \mathit{cens}_1(\mathcal{A}) = \{\ & \mathsf{WorksFor}(\mathsf{bob}, \mathsf{a}_1), \mathsf{Manages}(\mathsf{ann}, \mathsf{a}_2), \mathsf{WorksFor}(\mathsf{ann}, \mathsf{a}_2), \\ & \mathsf{BreachedBy}(\mathsf{a}_2, \mathsf{h}_2), \mathsf{GAgency}(\mathsf{a}_2)\ \}; \\ \mathit{cens}_4(\mathcal{A}) = \{\ & \mathsf{WorksFor}(\mathsf{bob}, \mathsf{a}_1), \mathsf{SServ}(\mathsf{a}_2), \mathsf{GAgency}(\mathsf{a}_2)\ \}. \end{aligned}$$

For the BCQ $q = \exists x, y\, \mathsf{OperatesIn}(x, y)$, we have that both $\mathcal{T} \cup \mathit{cens}_1(\mathcal{A}) \models q$ and $\mathcal{T} \cup \mathit{cens}_4(\mathcal{A}) \models q$ hold. ▽

Despite skeptically reasoning under all optimal GA censors is a natural approach for avoiding the randomness introduced by arbitrarily choosing one censor over the others, one can see that such an approach is intractable in data complexity (unless P = NP). Intuitively, this is due to the fact that the number of ways an ABox can be altered by optimal censors is exponential w.r.t. the size of the ABox itself. In fact, as shown in [83], the GA-Cens-Ent problem is coNP-hard already for BCQs.

Proposition 4 ([83, Thm. 6])**.** *GA-Cens-Ent for DL-Lite$_\mathcal{R}$ CQE specifications and BCQs is coNP-complete in data complexity.*

Obviously, from the theorem above it follows that GA-Cens-Ent is also coNP-hard for DL-Lite$_\mathcal{R}$ CQE specifications and BUCQs. Membership in coNP easily follows from the fact that the result shown in Theorem 6 of [83] is a consequence of Corollary 1 of the same work and on Theorem 4 of [80], which hold for FO sentences and BUCQs, respectively.

In order to overcome this computational issue, in the next section we propose a new semantics for CQE that soundly approximates the GA censor semantics.

3.3 IGA censor

Towards the identification of a practical setting, we propose a new semantically well-founded notion of GA censor which, on the one hand, makes, in the case of DL-Lite$_\mathcal{R}$, conjunctive query answering tractable (by approximating the skeptical reasoning), and, on the other hand, avoids arbitrary choices by being always unique.

The approximation we propose consists in considering a non-necessarily optimal GA censor corresponding to computing the intersection of all optimal GA censors for a CQE specification CQE.

Definition 8. [IGA censor] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification. The *Intersection-based GA censor* (*IGA censor*) for $\mathcal{E}$ is the function $\mathit{cens}_{\mathsf{IGA}}(\cdot)$ such that, for every ABox $\mathcal{A}$:

$$\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}) = \bigcap_{\mathit{cens}(\cdot) \in \mathit{optGACens}(\mathcal{E})} \mathit{cens}(\mathcal{A}).$$

Example 12. Recall the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ and the optimal GA censors $\mathit{cens}_1(\cdot)$ and $\mathit{cens}_4(\cdot)$ for $\mathcal{E}$ of Example 11. The IGA censor for $\mathcal{E}$ is the function $\mathit{cens}_{\mathsf{IGA}}(\cdot)$ such that $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}) = \mathit{cens}_1(\mathcal{A}) \cap \mathit{cens}_4(\mathcal{A})$ for each ABox $\mathcal{A}$, i.e.

$cens_{\mathsf{IGA}}(\mathcal{A})$ is the ABox obtained from $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ by removing (i) the ground atoms $\mathsf{Manages}(a,b)$, $\mathsf{WorksFor}(a,b)$, and $\mathsf{BreachedBy}(b,a)$ for all individuals $a,b \in \Gamma_I$ such that $\mathsf{SServ}(b) \in \mathcal{A}$ and (ii) the ground atoms $\mathsf{SServ}(b)$ such that $\mathsf{Manages}(a,b) \in \mathcal{A}$, or $\mathsf{WorksFor}(a,b) \in \mathcal{A}$ or $\mathsf{BreachedBy}(b,a) \in \mathcal{A}$ for all individuals $a,b \in \Gamma_I$.

As an example, consider the ABox $\mathcal{A}$ of Example 7. We have that $cens_{\mathsf{IGA}}(\mathcal{A}) = \{\mathsf{WorksFor}(\mathsf{bob}, \mathsf{a}_1),\ \mathsf{GAgency}(\mathsf{a}_1)\}$. ▽

As the above example shows, the IGA censor for a CQE specification $\mathcal{E}$ is not guaranteed to be an optimal GA censor. Notably, however, the following proposition shows that, for any CQE specification $\mathcal{E}$, the IGA censor exists and is unique.

Proposition 5. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification. The IGA censor $cens_{\mathsf{IGA}}(\cdot)$ for $\mathcal{E}$ exists, is unique and is a GA censor for $\mathcal{E}$.*

Proof. By looking at Definition 8, in order to prove that $cens_{\mathsf{IGA}}(\cdot)$ exists, it is enough to show the existence of at least an optimal GA censor $cens(\cdot)$ for $\mathcal{E}$. The uniqueness, then, directly follows from its definition. To see that there exists at least one optimal GA censor for $\mathcal{E}$, observe that (i) the function $cens_0(\cdot)$ such that $cens_0(\mathcal{A}) = \emptyset$ for each ABox $\mathcal{A}$ is a GA censor for $\mathcal{E}$, and (ii) each GA censor $cens(\cdot)$ for $\mathcal{E}$ is such that $cens(\mathcal{A})$ is a finite set of ground atoms for each ABox $\mathcal{A}$. This latter is guaranteed by the fact that $cens(\mathcal{A}) \subseteq cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ by definition, where $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ is a finite set of ground atoms since $\mathcal{A}$ is a finite set of ground atoms as well. So, (i) and (ii) imply the existence of at least one optimal GA censor for $\mathcal{E}$.

We now prove that $cens_{\mathsf{IGA}}(\cdot)$ is a GA censor for $\mathcal{E}$. Let $\mathcal{A}$ be any ABox. Note that $cens_{\mathsf{IGA}}(\mathcal{A}) \subseteq cens(\mathcal{A})$ for any optimal GA censor $cens(\cdot)$ for $\mathcal{E}$. Since, however, $\mathcal{T} \cup \mathcal{P} \cup cens(\mathcal{A})$ is consistent for any GA censor $cens(\cdot)$ for $\mathcal{E}$, we have that $\mathcal{T} \cup \mathcal{P} \cup cens_{\mathsf{IGA}}(\mathcal{A})$ is consistent as well due to the monotonicity of first-order logic. ∎

We show now that the IGA censor satisfies the indistinguishability property. First, we provide the counterparts of Lemma 1 and Lemma 2 for the IGA censor.

Lemma 3. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification and $\mathcal{A}$ be an ABox such that $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}$ is consistent. We have that $cens_{\mathsf{IGA}}(\mathcal{A}) = cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$.*

Proof. The thesis trivially follows from the definition of IGA censor and by Lemma 1. ∎

Lemma 4. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, and let $\mathcal{A}$ be an ABox. We have that $cens_{\mathsf{IGA}}(\mathcal{A}) = cl^{\mathcal{T}}_{\mathbf{GA}}(cens_{\mathsf{IGA}}(\mathcal{A}))$.*

Proof. Note that $cens_{\mathsf{IGA}}(\mathcal{A}) \subseteq cl^{\mathcal{T}}_{\mathbf{GA}}(cens(\mathcal{A}))$ trivially holds. We now prove that $cl^{\mathcal{T}}_{\mathbf{GA}}(cens(\mathcal{A})) \subseteq cens_{\mathsf{IGA}}(\mathcal{A})$.

Towards a contradiction, suppose this is not the case, i.e. there exists a $\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(cens_{\mathsf{IGA}}(\mathcal{A}))$ such that $\alpha \notin cens_{\mathsf{IGA}}(\mathcal{A})$. Since $\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(cens_{\mathsf{IGA}}(\mathcal{A}))$, then there exists in $cens_{\mathsf{IGA}}(\mathcal{A})$ a set of facts $\mathcal{F}$ such that $\mathcal{T} \cup \mathcal{F} \models \alpha$. Moreover, since $cens_{\mathsf{IGA}}(\mathcal{A}) \subseteq cens(\mathcal{A})$ for each optimal **GA** censor $cens(\cdot)$ for $\mathcal{E}$, we also have that $\mathcal{F} \subseteq cens(\mathcal{A})$, and so, given the monotonicity of first-order-logic, we have that $\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(cens(\mathcal{A}))$ for each optimal censor $cens(\cdot)$ for $\mathcal{E}$. Therefore, by Lemma 2, we have also that $\alpha \in cens(\mathcal{A})$, for each optimal **GA** censor $cens(\cdot)$ for $\mathcal{E}$, from

which it clearly holds that $\alpha \in \mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$. This contradicts the initial assumption that $\alpha \notin \mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$. Thus, it must be that $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}) = \mathit{cl}^{\mathcal{T}}_{\mathbf{GA}}(\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}))$. ∎

We are now ready to show that the indistinguishability property holds for the IGA censor.

Proposition 6. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification. The IGA censor $\mathit{cens}_{\mathsf{IGA}}(\cdot)$ for $\mathcal{E}$ satisfies the indistinguishability property.*

Proof. We prove that the ABox $\mathcal{A}'$ of Definition 6 is the set $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$ itself, i.e. we show that $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$ is an ABox such that $\mathcal{T} \cup \mathcal{P} \cup \mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$ is consistent and $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}) = \mathit{cens}_{\mathsf{IGA}}(\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}))$ for each ABox $\mathcal{A}$.

Consider any ABox $\mathcal{A}$. By Lemma 4, we have that $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}) = \mathit{cl}^{\mathcal{T}}_{\mathbf{GA}}(\mathit{cens}(\mathcal{A}))$. Furthermore, since $\mathcal{T} \cup \mathcal{P} \cup \mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$ is consistent, then Lemma 3 implies that $\mathit{cens}_{\mathsf{IGA}}(\mathit{cens}_{\mathsf{IGA}}(\mathcal{A})) = \mathit{cl}^{\mathcal{T}}_{\mathbf{GA}}(\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}))$. It follows that $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}) = \mathit{cl}^{\mathcal{T}}_{\mathbf{GA}}(\mathit{cens}_{\mathsf{IGA}}(\mathcal{A})) = \mathit{cens}_{\mathsf{IGA}}(\mathit{cens}_{\mathsf{IGA}}(\mathcal{A}))$, as required. ∎

We point out that we could in principle have considered a counterpart of Definition 8 also for CQ censors. However, it is easy to see that such a censor might still not satisfy the indistinguishability property. In Example 10, for instance, the resulting "intersection-based" function would coincide with the unique CQ censor for the given CQE specification.

3.3.1 Query answering under IGA censor

In this section, we focus on DL-Lite$_{\mathcal{R}}$ CQE specifications and BUCQs as queries, and we show that query answering based on the IGA censor is reducible to the evaluation of an FO query over the ABox, i.e. it is FO rewritable and therefore in AC^0 in data complexity.

Below, we provide the general decision problem we focus on.

Definition 9. [IGA-Cens-Ent] Given as input a CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, an ABox $\mathcal{A}$, and a Boolean query q, IGA-Cens-Ent is the problem of deciding whether $\mathcal{T} \cup \mathit{cens}_{\mathsf{IGA}}(\mathcal{A}) \models q$. In case this holds, we write $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{IGA}} q$.

The following proposition, whose proof is straightforward, says that the above entailment semantics is a sound approximation of the skeptical reasoning approach under GA censors.

Proposition 7. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, and let q be a Boolean query. If $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{IGA}} q$ holds, then $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{GA}} q$ holds too.*

The next example shows that, in general, the converse of Proposition 7 does not hold.

Example 13. We refer to the same CQE specification and ABox of Example 11. While we have that $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{GA}} q$ holds, observe that $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{IGA}} q$ does not, because $\mathcal{T} \cup \mathit{cens}_{\mathsf{IGA}}(\mathcal{A}) \not\models q$. ▽

In what follows on this section, we show that IGA-Cens-Ent is FO-rewritable in our considered scenario, that is, for every DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}$ and BUCQ q, one can effectively compute an FO query q_r such that, for every ABox $\mathcal{A}$, $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{IGA}} q$ iff q_r evaluates to true in $\mathcal{A}$. We call such a query q_r the *IGA reformulation* of q with respect to $\mathcal{E}$.

The intuition behind our rewriting algorithm is as follows. For a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, given any ABox $\mathcal{A}$, we want to filter out facts from $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ that together with the TBox $\mathcal{T}$ lead to the violation of some denial in the policy $\mathcal{P}$. At the same time, we want this elimination of facts to be done in a minimal way, according to our definition of the IGA censor. Thus, only "really dangerous" facts have to be dropped from $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. Let us first formally define such a minimal set of facts violating the policy, that we call *secrets*.

Definition 10. [Secret] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification and $\mathcal{A}$ be an ABox. A set of facts $\mathcal{F} \subseteq cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ is a *secret for $\mathcal{E}$ and $\mathcal{A}$* if:

(*i*) $\mathcal{T} \cup \mathcal{P} \cup \mathcal{F}$ is inconsistent, and

(*ii*) there is no set $\mathcal{F}'$ of ground atoms such that $\mathcal{F}' \subset \mathcal{F}$ and $\mathcal{T} \cup \mathcal{P} \cup \mathcal{F}'$ is inconsistent.

Given a CQE specification $\mathcal{E}$ and an ABox $\mathcal{A}$, we denote by $\mathit{secrets}(\mathcal{E}, \mathcal{A})$ the set of secrets for $\mathcal{E}$ and $\mathcal{A}$.

Example 14. Consider the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ and ABox $\mathcal{A}$ of the previous examples. One can see that $\mathit{secrets}(\mathcal{E}, \mathcal{A}) = \{\{\mathsf{SServ}(\mathsf{a}_2),\ \mathsf{Manages}(\mathsf{ann}, \mathsf{a}_2)\},$ $\{\mathsf{SServ}(\mathsf{a}_2), \mathsf{WorksFor}(\mathsf{ann}, \mathsf{a}_2)\},\ \{\mathsf{SServ}(\mathsf{a}_2),\ \mathsf{BreachedBy}(\mathsf{a}_2, \mathsf{h}_2)\}\}$. These are indeed the only three minimal sets $\mathcal{F} \subseteq cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ such that $\mathcal{T} \cup \mathcal{P} \cup \mathcal{F}$ is inconsistent. ▽

The next key proposition shows that, for DL-Lite$_\mathcal{R}$ CQE specifications $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, given an ABox $\mathcal{A}$, the IGA censor for $\mathcal{E}$ returns all and only those ground atoms in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ that does not occur in any secret for $\mathcal{E}$ and $\mathcal{A}$.

Proposition 8. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{R}$ CQE specification, let $\mathcal{A}$ be an ABox, and let $\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. We have that $\alpha \notin \mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$ iff $\alpha \in \mathcal{F}$ for some $\mathcal{F} \in \mathit{secrets}(\mathcal{E}, \mathcal{A})$.*

Proof. Note that a ground atom $\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ does not belong to $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$ iff there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\alpha \notin \mathit{cens}(\mathcal{A})$. So it is sufficient to show that $\alpha \notin \mathit{cens}(\mathcal{A})$ for some optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ iff there exists a secret $\mathcal{F}$ in $\mathit{secrets}(\mathcal{E}, \mathcal{A})$ such that $\alpha \in \mathcal{F}$.

($\Rightarrow$:) Suppose, by way of contradiction, that α does not belong to any secret in $\mathit{secrets}(\mathcal{E}, \mathcal{A})$. It follows that, for any optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$, $\mathcal{T} \cup \mathcal{P} \cup \mathit{cens}(\mathcal{A}) \cup \{\alpha\}$ is consistent (otherwise, one can see that α would belong to some secret in $\mathit{secrets}(\mathcal{E}, \mathcal{A})$) and so $\mathit{cens}(\cdot)$ is not an optimal GA censor for $\mathcal{E}$, from which the contradiction that $\mathit{cens}(\cdot)$ is an optimal GA censor for $\mathcal{E}$. Indeed, the function $\mathit{cens}'(\cdot)$ with $\mathit{cens}'(\mathcal{A}') = \mathit{cens}(\mathcal{A}')$ for each ABox $\mathcal{A}' \neq \mathcal{A}$ and $\mathit{cens}'(\mathcal{A}) = \mathit{cens}(\mathcal{A}) \cup \{\alpha\}$ would be a GA censor for $\mathcal{E}$ that is more informative than $\mathit{cens}(\cdot)$.

($\Leftarrow$:) Suppose that $\alpha \in \mathcal{F}$ for some secret $\mathcal{F} \in \mathit{secrets}(\mathcal{E}, \mathcal{A})$. We now show that there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\alpha \notin \mathit{cens}(\mathcal{A})$. From

Definition 10, we can derive that $\mathcal{T} \cup \mathcal{P} \cup (\mathcal{F} \setminus \{\alpha\})$ is consistent. It follows that there exists at least an optimal GA censor $cens(\cdot)$ for $\mathcal{E}$ such that $(\mathcal{F} \setminus \{\alpha\}) \subseteq cens(\mathcal{A})$. ∎

Identifying the ground atoms occurring in some secret is easier if we can reason on each denial in the policy in isolation. Unfortunately, this may not always be possible even in very simple cases. Consider, for instance, the CQE specification $\mathcal{E} = \langle \emptyset, \mathcal{P} \rangle$, with $\mathcal{P} = \{\forall x\,(A(x) \wedge B(x) \rightarrow \bot), \forall x\,(A(x) \rightarrow \bot)\}$. The set $\{A(d), B(d)\}$ of ground atoms occurring in an ABox $\mathcal{A}$ is a minimal set violating the first denial, but is not a secret for $\mathcal{E}$ and $\mathcal{A}$ since its subset $\{A(d)\}$ belongs to $secrets(\mathcal{E}, \mathcal{A})$.

To be able to solve this issue and treat denials separately, we now introduce the notion of *extended denial assertion* (or simply *extended denial*), which is a formula of the form $\forall \mathbf{x}\,(\gamma(\mathbf{x}) \rightarrow \bot)$ such that $\exists \mathbf{x}\,\gamma(\mathbf{x})$ is a BCQ$^{\neq}$. Then, an *extended policy* is a finite set of extended denials. Moreover, in the rest of this section, we call *non-extended denial* a denial as defined in Section 3.1.

Definition 11. [Non-redundant representation] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification. An extended policy $\mathcal{P}'$ is a *non-redundant representation of $\mathcal{P}$ with respect to $\mathcal{T}$* if the following conditions hold:

(*i*) for every ABox $\mathcal{A}$, we have that $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}$ is inconsistent iff $\mathcal{P}' \cup \mathcal{A}$ is inconsistent;

(*ii*) for every $\delta \in \mathcal{P}'$, ABox $\mathcal{A}$, and minimal (with respect to set inclusion) set of ground atoms $\mathcal{F} \subseteq cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ such that $\{\delta\} \cup \mathcal{F}$ is inconsistent, we have that $\mathcal{F} \in secrets(\mathcal{E}, \mathcal{A})$.

One might think that, given a CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, computing an extended policy $\mathcal{P}'$ that is a non-redundant representation of $\mathcal{P}$ with respect to $\mathcal{T}$ means simply eliminating from $\mathcal{P}$ each denial δ such that $\mathcal{T} \cup \mathcal{P} \setminus \{\delta\} \models \delta$. In fact, only eliminating denials that are (fully) logically inferred by other denials (and the TBox) is not sufficient, since some redundancies can occur for specific instantiations of the denials. For example, consider the CQE specification $\langle \mathcal{T}, \mathcal{P} \rangle$, where $\mathcal{T} = \emptyset$ and $\mathcal{P} = \{\delta_1, \delta_2\}$ with $\delta_1 = \forall x, y\,(Q(x, y) \wedge C(y) \rightarrow \bot)$ and $\delta_2 = \forall z\,(Q(z, z) \rightarrow \bot)$. Despite the fact that $\mathcal{T} \cup \{\delta_2\} \not\models \delta_1$, for the particular instantiation in which $x = y$, giving rise to denial $\delta'_1 = \forall x\,(Q(x, x) \wedge C(x) \rightarrow \bot)$, we have that $\mathcal{T} \cup \{\delta_2\} \models \delta'_1$. This implies that, for instance, given an ABox $\mathcal{A}$ containing the ABox assertions $Q(a, a)$ and $C(a)$, although $\mathcal{F} = \{Q(a, a), C(a)\}$ is a minimal violation of δ_1, $\mathcal{F}$ is not a secret for $\mathcal{E}$ and $\mathcal{A}$ since $\{Q(a, a)\}$ is already a secret for $\mathcal{E}$ and $\mathcal{A}$. An extended policy that would be a non-redundant representation of $\mathcal{P}$ with respect to $\mathcal{T}$ is $\mathcal{P}' = \{\delta''_1, \delta_2\}$, where $\delta''_1 = \forall x, y\,(Q(x, y) \wedge C(y) \wedge x \neq y \rightarrow \bot)$.

Actually, given a DL-Lite$_{\mathcal{R}}$ TBox $\mathcal{T}$ and a policy $\mathcal{P}$, in order to compute an extended policy $\mathcal{P}'$ that is a non-redundant representation of $\mathcal{P}$ with respect to $\mathcal{T}$, it is possible to use the same technique illustrated in [82] in the context of Consistent Query Answering. In that work the authors introduce an algorithm called *MinUnsatQuery*, which serves an analogous purpose. We point out that *MinUnsatQuery* takes as input a set composed of DL-Lite$_{\mathcal{R}}$ TBox axioms and denial assertions. In what follows, given an extended denial $\delta = \forall \mathbf{x}\,(\gamma(\mathbf{x}) \rightarrow \bot)$, we denote by q_δ the BCQ$^{\neq}$ associated to δ, i.e. $q_\delta = \exists \mathbf{x}\,\gamma(\mathbf{x})$, and, on the other direction, given a BCQ$^{\neq}$ $q = \exists \mathbf{x}\,\gamma(\mathbf{x})$, we let $\delta_q = \forall \mathbf{x}\,(\gamma(\mathbf{x}) \rightarrow \bot)$. Now, given a DL-Lite$_{\mathcal{R}}$ TBox

$\mathcal{T}$ and a policy $\mathcal{P}$, we define the function *MinPolicy* as follows: $\mathit{MinPolicy}(\mathcal{P},\mathcal{T}) = \{\delta_q \mid q \in \mathit{MinUnsatQuery}(\mathcal{T}^p \cup \mathcal{P})\}$, where $\mathcal{T}^p$ is the "positive" part of the DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$, i.e. the set of axioms obtained from $\mathcal{T}$ by removing all the disjointness axioms occurring in it.

Example 15. Consider the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ of the running example. We have that:

$$\begin{aligned}\mathit{MinPolicy}(\mathcal{P},\mathcal{T}) = \{\ & \forall x,y\,(\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y,x) \wedge x \neq y \rightarrow \bot),\\ & \forall x,y\,(\mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(y,x) \wedge x \neq y \rightarrow \bot),\\ & \forall x,y\,(\mathsf{SServ}(x) \wedge \mathsf{Manages}(y,x) \wedge x \neq y \rightarrow \bot),\\ & \forall x\,(\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(x,x) \rightarrow \bot),\\ & \forall x\,(\mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(x,x) \rightarrow \bot),\\ & \forall x\,(\mathsf{SServ}(x) \wedge \mathsf{Manages}(x,x) \rightarrow \bot)\ \}.\end{aligned}$$

Anyway, for the sake of conciseness, in the subsequent examples we always refer to the following simplified version of the above set:

$$\begin{aligned}\{\ & \forall x,y\,(\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y,x) \rightarrow \bot),\\ & \forall x,y\,(\mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(y,x) \rightarrow \bot),\\ & \forall x,y\,(\mathsf{SServ}(x) \wedge \mathsf{Manages}(y,x) \rightarrow \bot)\ \}.\end{aligned}$$

Observe that such a set is still a non-redundant representation of $\mathcal{P}$ w.r.t. $\mathcal{T}$. ▽

Proposition 9. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ TBox and $\mathcal{P}$ be a policy. The extended policy returned by MinPolicy$(\mathcal{P},\mathcal{T})$ is a non-redundant representation of $\mathcal{P}$ with respect to $\mathcal{T}$.*

Proof. The thesis follows immediately from Definition 11 and from Lemma 6 and Lemma 7 of [82]. ∎

We are now ready to provide our query rewriting technique. In what follows, without loss of generality, we assume that, given a DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$ and a policy $\mathcal{P}$, the output of $\mathit{MinPolicy}(\mathcal{P},\mathcal{T})$ is an extended policy such that in each extended denial the arguments of the various atoms are always variables different to one another (the presence of the same variable or of constants can be indeed expressed through equalities).

The basic idea is to exploit Proposition 8 for individuating, given a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ and an ABox $\mathcal{A}$, those atoms that can be involved in the evaluation of a BUCQ over $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$. This is done by discarding those atoms in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ that occur in some secret for $\mathcal{E}$ and $\mathcal{A}$, that is, as follows from Proposition 9, the atoms that participate in the image of at least one BCQ$^{\neq}$ associated to some denial in $\mathit{MinPolicy}(\mathcal{P},\mathcal{T})$.

Given a predicate atom α and an FO sentence ϕ, we denote by $\mathit{findSub}(\alpha,\phi)$ the set of substitutions σ of the variables occurring in any atom $\beta \in \phi$ to the terms occurring in α such that $\sigma(\beta) = \alpha$. For an atom α and a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, we define $\mathit{Discl}(\alpha,\mathcal{E})$ as follows:

$$\mathit{Discl}(\alpha,\mathcal{E}) = \alpha \wedge \left(\bigwedge_{\substack{\delta \in \mathit{MinPolicy}(\mathcal{P},\mathcal{T}),\\ \sigma \in \mathit{findSub}(\alpha,q_\delta)}} \forall \mathbf{w}\ (\neg\sigma(q_\delta)) \right),$$

where $\mathbf{w}$ contains all the variables occurring in $\sigma(q_\delta)$ but not in α.

Example 16. Consider again the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ of the running example, and let $\alpha = \mathsf{BreachedBy}(x, y)$. We have that $\mathit{Discl}(\alpha, \mathcal{E}) = \mathsf{BreachedBy}(x, y) \wedge \forall w\,(\neg(\mathsf{BreachedBy}(x, w) \wedge \mathsf{SServ}(x)))$. A shorter, yet equivalent, version of such a formula is $\mathsf{BreachedBy}(x, y) \wedge \neg\mathsf{SServ}(x)$. ▽

Given a BUCQ q and a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}$, we define the FO sentence $\mathit{DQ}(q, \mathcal{E})$ as follows:

$$\mathit{DQ}(q, \mathcal{E}) = \bigvee_{\exists \mathbf{x}\, \gamma \in q} \left(\exists \mathbf{x} \bigwedge_{\alpha \in \mathit{predAt}(\gamma)} \mathit{Discl}(\alpha, \mathcal{E}) \right).$$

Example 17. Consider the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ and the query q described in previous examples. We have that $\mathit{PerfectRef}(q, \mathcal{T}) = \exists x, y\, \mathsf{OperatesIn}(x, y) \vee \exists x, y\, \mathsf{BreachedBy}(x, y) \vee \exists x\, \mathsf{SServ}(x)$. Moreover, one can verify that:

$$\begin{aligned} \mathit{DQ}(\mathit{PerfectRef}(q, \mathcal{T}), \mathcal{E}) = {} & \exists x, y\, \mathsf{OperatesIn}(x, y) \vee \\ & \exists x, y\, (\mathsf{BreachedBy}(x, y) \wedge \neg\mathsf{SServ}(x)) \vee \\ & \exists x\, (\mathsf{SServ}(x) \wedge \forall y\, (\neg\mathsf{BreachedBy}(x, y)) \wedge \\ & \quad \forall y\, (\neg\mathsf{Manages}(y, x)) \wedge \forall y\, (\neg\mathsf{WorksFor}(y, x))) \end{aligned}$$ ▽

Given a DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$ and an FO query ϕ, we define $\mathit{AtomRewr}(\phi, \mathcal{T})$ as the FO query obtained from ϕ by replacing every atom α occurring in ϕ with its "TBox-expansion" $\mathit{AtomRewr}(\alpha, \mathcal{T})$, defined as follows:

$$\mathit{AtomRewr}(\alpha, \mathcal{T}) = \begin{cases} \bigvee_{\mathcal{T} \models D \sqsubseteq C} D(t) \vee & \\ \bigvee_{\mathcal{T} \models \exists R \sqsubseteq C} (\exists x\, R(t, x)) \vee & \text{if } \alpha = C(t) \\ \bigvee_{\mathcal{T} \models \exists R^- \sqsubseteq C} (\exists x\, R(x, t)), & \\ \bigvee_{\mathcal{T} \models S \sqsubseteq R} S(t_1, t_2) \vee & \\ \bigvee_{\mathcal{T} \models S^- \sqsubseteq R} S(t_2, t_1), & \text{if } \alpha = R(t_1, t_2) \end{cases}$$

Finally, for a BUCQ q and a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, we define the following function returning an FO sentence:

$$\mathit{DRew}(q, \mathcal{E}) = \mathit{AtomRewr}(\mathit{DQ}(\mathit{PerfectRef}(q, \mathcal{T}), \mathcal{E}), \mathcal{T}).$$

Example 18. Consider the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ and the query q of the running example. One can see that:

$$\begin{aligned} \mathit{DRew}(q, \mathcal{E}) = {} & \exists x, y\, \mathsf{OperatesIn}(x, y) \vee \\ & \exists x, y\, (\mathsf{BreachedBy}(x, y) \wedge \neg\mathsf{SServ}(x)) \vee \\ & \exists x\, (\mathsf{SServ}(x) \wedge \forall y\, (\neg\mathsf{BreachedBy}(x, y)) \wedge \forall y\, (\neg\mathsf{Manages}(y, x)) \wedge \\ & \quad \forall y\, (\neg(\mathsf{WorksFor}(y, x) \vee \mathsf{Manages}(y, x)))) \end{aligned}$$ ▽

We are now ready to prove that, for DL-Lite$_\mathcal{R}$ CQE specifications $\mathcal{E}$ and BUCQs q, the decision problem IGA-Cens-Ent can always be solved by checking whether $\mathit{DRew}(q, \mathcal{E})$ evaluates to true in $\mathcal{A}$. In other terms, we prove that the problem is FO rewritable. We start with the next lemma, which immediately follows from the definition of $\mathit{cl}^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ and $\mathit{AtomRewr}(\phi, \mathcal{T})$.

Lemma 5. *Let $\mathcal{T}$ be a DL-Lite$_\mathcal{R}$ CQE specification, let $\mathcal{A}$ be an ABox, and let ϕ be an FO query. Then, ϕ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ iff $AtomRewr(\phi, \mathcal{T})$ evaluates to true in $\mathcal{A}$.*

On the basis of Proposition 1 and Lemma 5, we prove FO-rewritability of IGA-Cens-Ent for DL-Lite$_\mathcal{R}$ CQE specifications and BUCQs.

Theorem 2. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{R}$ CQE specification, and let q be a BUCQ. For every ABox $\mathcal{A}$, we have that $\mathcal{T} \cup cens_{\mathsf{IGA}}(\mathcal{A}) \models q$ iff $DRew(q, \mathcal{E})$ evaluates to true in $\mathcal{A}$.*

Proof. By exploiting Proposition 1 and Lemma 5, to prove the thesis of the theorem note that it is enough to show the following: given a BUCQ Q, we have that Q evaluates to true in $cens_{\mathsf{IGA}}(\mathcal{A})$ iff $DQ(Q, \mathcal{E})$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$.

First, given any BUCQ $Q = q_1 \vee \ldots \vee q_n$ and ABox $\mathcal{A}'$, we note that Q evaluates to true in $\mathcal{A}'$ iff q_i does, for some i such that $1 \leq i \leq n$. Then, from the definition of $DQ(\cdot, \cdot)$ it follows that, for any BCQ $q \in Q$, it is enough to prove that q evaluates to true in $cens_{\mathsf{IGA}}(\mathcal{A})$ iff $DQ(q, \mathcal{E})$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. Note that, since q is a BCQ, then $DQ(q, \mathcal{E}) = \exists \mathbf{x} \bigwedge_{\alpha \in predAt(q)} Discl(\alpha, \mathcal{E})$, where $\mathbf{x}$ are the variables of q.

($\Rightarrow$:) Suppose $\exists \mathbf{x} \bigwedge_{\alpha \in q} Discl(\alpha, \mathcal{E})$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. By construction of $Discl(\cdot, \cdot)$, this means that there is at least an image M of q in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ such that $\exists \mathbf{x}_\alpha \, Discl(\alpha, \mathcal{E})$ evaluates to true in M for any atom $\alpha \in q$, where $\mathbf{x}_\alpha \subseteq \mathbf{x}$ are the variables occurring in the atom α. We denote by M_α the image (a ground atom) of α in M. Now, Proposition 9 guarantees that there is no $\mathcal{F} \in secrets(\mathcal{E}, \mathcal{A})$ such that $M_\alpha \in \mathcal{F}$, otherwise this would easily contradict the fact that $\exists \mathbf{x}_\alpha \, Discl(\alpha, \mathcal{E})$ evaluates to true in M. It follows that $M = \bigcup_{\alpha \in q} M_\alpha$ is such that there is no atom in M that belongs to some secret for $\mathcal{E}$ and $\mathcal{A}$. Now, by Proposition 8, it directly follows that $M \subseteq cens_{\mathsf{IGA}}(\mathcal{A})$. Since q obviously evaluates to true in M, it evaluates to true in $cens_{\mathsf{IGA}}(\mathcal{A})$ as well by monotonicity of BCQ evaluation.

($\Leftarrow$:) Suppose now that q evaluates to true in $cens_{\mathsf{IGA}}(\mathcal{A})$. Consider each atom $\alpha \in q$. Since, by Proposition 8, we know that all the ground atoms in $cens_{\mathsf{IGA}}(\mathcal{A})$ do not belong to any secrets for $\mathcal{E}$ and $\mathcal{A}$, by Proposition 9 and by construction of $Discl(\cdot, \cdot)$, we easily derive that $\exists \mathbf{x}_\alpha \, Discl(\alpha, \mathcal{E})$ evaluates to true in $cens_{\mathsf{IGA}}(\mathcal{A})$ as well. Thus, we immediately obtain that $\exists \mathbf{x}_q \bigwedge_{\alpha \in q} Discl(\alpha, \mathcal{E})$ evaluates to true in $cens_{\mathsf{IGA}}(\mathcal{A})$, as required. ∎

Example 19. Consider the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, the ABox $\mathcal{A}$, and the query q of the running example, and consider the IGA reformulation $DRew(q, \mathcal{E})$ of q with respect to $\mathcal{E}$ illustrated in Example 18. Note that $DRew(q, \mathcal{E})$ evaluates to false in $\mathcal{A}$. Indeed, as observed in Example 13, we have that $\mathcal{T} \cup cens_{\mathsf{IGA}}(\mathcal{A}) \not\models q$. ▽

Provided that $DRew(\cdot, \cdot)$ first rewrites the input query via *PerfectRef*, one may wonder whether rewriting atom by atom through the $AtomRewr(\cdot, \cdot)$ function is necessary. We then show a simple example illustrating how performing such step is indeed essential for ensuring correctness of our approach. Consider a TBox $\mathcal{T} = \{A \sqsubseteq B\}$, an ABox $\mathcal{A} = \{A(o)\}$ and a policy $\mathcal{P} = \{\forall x \, (A(x) \rightarrow \bot)\}$. It is easy to see that there is only one optimal GA censor $cens(\cdot)$ for $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, which

is such that $cens(\mathcal{A}) = \{B(o)\}$. Thus, also the IGA censor for $\mathcal{E}$ is such that $cens_{\mathsf{IGA}}(\mathcal{A}) = \{B(o)\}$. Now, consider a BCQ $q = \exists x\, B(x)$, which clearly evaluates to true in $cens_{\mathsf{IGA}}(\mathcal{A})$. Rewriting q without using the $AtomRewr(\cdot,\cdot)$ function would result in the following FO query: $\phi = \exists x\, B(x) \vee \exists x\, (A(x) \wedge \neg A(x))$, whose evaluation in $\mathcal{A}$ returns false. On the other hand, by adopting the $AtomRewr(\cdot,\cdot)$ function we get the FO query $AtomRewr(\phi, \mathcal{T}) = \exists x\, (B(x) \vee A(x)) \vee \exists x\, (A(x) \wedge \neg A(x))$, which evaluates to true in $\mathcal{A}$.

The corollary below follows from Theorem 2 and the fact that evaluating an FO query over an ABox is in AC^0 in the size of the ABox (i.e. in data complexity).

Corollary 1. *IGA-Cens-Ent for DL-Lite$_\mathcal{R}$ CQE specifications and BUCQs is in AC^0 in data complexity.*

Chapter 4

Prioritized controlled query evaluation

Answering queries under the IGA semantics may be too restrictive. This is due to the fact that any ground atom belonging to any secret is not included in the set returned by the IGA censor. In this chapter, we present an extension of the CQE framework in which one can specify a priority relation over ontology predicates. These priorities can be used for inducing a choice between facts belonging to the same secret, in order to reveal some facts that would instead be kept undisclosed, thus improving the throughput of answers to user queries with respect to the IGA semantics.

4.1 Framework extension and common priority-based semantics

Given a TBox $\mathcal{T}$, a priority relation $\succ$ over $\mathcal{T}$ is an acyclic binary relation between predicates occurring in the signature of $\mathcal{T}$, i.e. $\succ \subseteq \mathit{pred}(\mathcal{T}) \times \mathit{pred}(\mathcal{T})$.

Definition 12. [Prioritized CQE specification] Let $\mathcal{L}_\mathcal{T}$ be a DL. A *prioritized $\mathcal{L}_\mathcal{T}$ CQE specification* $\mathcal{E}_\succ$ is a triple $\langle \mathcal{T}, \mathcal{P}, \succ \rangle$, such that $\langle \mathcal{T}, \mathcal{P} \rangle$ is an $\mathcal{L}_\mathcal{T}$ CQE specification and $\succ$ is a priority relation over $\mathcal{T}$.

Similarly to what done for (non-prioritized) CQE specifications, we omit $\mathcal{L}_\mathcal{T}$ for definitions and results applying to any DL.

Example 20. Suppose that, if we know that a person p works for a secret service s (both $\mathsf{WorksFor}(p, s)$ and $\mathsf{SServ}(s)$ hold), we prefer to disclose the fact that p works for s ($\mathsf{WorksFor}(p, s)$) rather than the fact that s is a secret service ($\mathsf{SServ}(s)$). This is indicated with $(\mathsf{WorksFor},\ \mathsf{SServ}) \in \succ$.

In the following examples, we refer to the prioritized DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$, where $\langle \mathcal{T}, \mathcal{P} \rangle$ is the DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}$ of Example 4 and $\succ = \{(\mathsf{WorksFor},\ \mathsf{SServ}), (\mathsf{Manages},\ \mathsf{SServ})\}$. ▽

The definitions of GA censor, optimal GA censor, IGA censor, GA-Cens-Ent, and IGA-Cens-Ent apply also to a prioritized CQE specification (e.g. given one

such specification $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$, the function $cens(\cdot)$ is a GA censor for $\mathcal{E}_\succ$ if it is a GA censor for the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$). We also use for prioritized CQE specifications the same notations introduced for (non-prioritized) CQE specifications, with the same meaning. In particular, we denote by $optGACens(\mathcal{E}_\succ)$ the set of optimal GA censors for $\mathcal{E}$ and, given an ABox $\mathcal{A}$, we write $secrets(\mathcal{E}_\succ, \mathcal{A})$ for actually meaning $secrets(\mathcal{E}, \mathcal{A})$.

We now exploit the priority relation to induce a partial order over censors. We consider two optimality notions introduced by [111] in the context of consistent query answering over databases, and recently adopted in [25] for repairing inconsistent prioritized DL ontologies.[1] Whereas the priority relations considered in this thesis are intentional, i.e. between ontology predicates, priorities considered in [111, 25] are between (conflicting) facts. However, intensional priorities straightforwardly induce priorities over facts: given a TBox $\mathcal{T}$, a priority relation $\succ$ over $\mathcal{T}$, an ABox $\mathcal{A}$, and two assertions $S_1(\mathbf{n})$ and $S_2(\mathbf{m})$ in $\mathcal{A}$, we have that $S_1(\mathbf{n}) \succ S_2(\mathbf{m})$ if $S_1 \succ S_2$.

Below we take the definitions of Pareto- and Globally-optimal repair from [25] and adapt them to our framework.

Definition 13. [Pareto/Globally-optimal censor] Let $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$ be a prioritized CQE specification, $\mathcal{A}$ be an ABox, and $cens(\cdot) \in optGACens(\mathcal{E}_\succ)$. We say that an ABox $\mathcal{A}' \subseteq cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, such that $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}'$ is consistent, is:

- a *Pareto improvement* of $cens(\mathcal{A})$ with respect to $\mathcal{E}_\succ$ if there exists an assertion $\alpha' \in \mathcal{A}' \setminus cens(\mathcal{A})$ such that, for every assertion $\alpha \in cens(\mathcal{A}) \setminus \mathcal{A}'$, we have that $\alpha' \succ \alpha$ and $\{\alpha, \alpha'\} \subseteq \mathcal{F}$ for some $\mathcal{F} \in secrets(\mathcal{E}_\succ, \mathcal{A})$;

- a *Global improvement* of $cens(\mathcal{A})$ with respect to $\mathcal{E}_\succ$ if $\mathcal{A}' \neq cens(\mathcal{A})$ and, for every assertion $\alpha \in cens(\mathcal{A}) \setminus \mathcal{A}'$, there exists an assertion $\alpha' \in \mathcal{A}' \setminus cens(\mathcal{A})$ such that $\alpha' \succ \alpha$ and $\{\alpha, \alpha'\} \subseteq \mathcal{F}$ for some $\mathcal{F} \in secrets(\mathcal{E}_\succ, \mathcal{A})$.

Then, $cens(\cdot)$ is a *Pareto- (resp., Globally-)optimal censor* for $\mathcal{E}_\succ$ if there exists no other GA censor $cens'(\cdot)$ for $\mathcal{E}_\succ$ such that, for each ABox $\mathcal{A}$, either $cens'(\mathcal{A}) = cens(\mathcal{A})$ or $cens'(\mathcal{A})$ is a Pareto (resp., Global) improvement of $cens(\mathcal{A})$ with respect to $\mathcal{E}_\succ$.

We denote by $PCens(\mathcal{E}_\succ)$ (resp., $GCens(\mathcal{E}_\succ)$) the set of all Pareto- (resp., Globally-) optimal censors for $\mathcal{E}_\succ$. It is easy to see that $GCens(\mathcal{E}_\succ) \subseteq PCens(\mathcal{E}_\succ) \subseteq optGACens(\mathcal{E}_\succ)$ for every prioritized CQE specification $\mathcal{E}_\succ$, analogous to the containment between Pareto- and Globally-optimal repairs given in [111]. Also, if $\succ$ is empty, then $PCens(\mathcal{E}_\succ) = GCens(\mathcal{E}_\succ) = optGACens(\mathcal{E}_\succ)$.

As done for GA censors (see Definition 8), we define intersection-based versions of both Pareto- and Globally-optimal censors.

Definition 14. [IP/IG censor] Given a prioritized CQE specification $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$, we call:

[1] As discussed in the introduction, there is a significant connection between inconsistency-tolerant query answering and CQE.

- *Intersection Pareto (IP) censor* for $\mathcal{E}_{\succ}$ the function $cens_{\mathsf{IP}}(\cdot)$ such that, for every ABox $\mathcal{A}$:

$$cens_{\mathsf{IP}}(\mathcal{A}) = \bigcap_{cens(\cdot) \in PCens(\mathcal{E}_{\succ})} cens(\mathcal{A})$$

- *Intersection Global (IG) censor* for $\mathcal{E}_{\succ}$ the function $cens_{\mathsf{IG}}(\cdot)$ such that, for every ABox $\mathcal{A}$:

$$cens_{\mathsf{IG}}(\mathcal{A}) = \bigcap_{cens(\cdot) \in GCens(\mathcal{E}_{\succ})} cens(\mathcal{A})$$

Obviously, $cens_{\mathsf{IP}}(\mathcal{A}) \subseteq cens_{\mathsf{IG}}(\mathcal{A})$ for each ABox $\mathcal{A}$. Also, if $\succ$ is empty, then, since $PCens(\mathcal{E}_{\succ}) = GCens(\mathcal{E}_{\succ}) = optGACens(\mathcal{E}_{\succ})$, we have that $cens_{\mathsf{IP}}(\cdot) = cens_{\mathsf{IG}}(\cdot) = cens_{\mathsf{IGA}}(\cdot)$.

Given as input a prioritized CQE specification $\mathcal{E}_{\succ}$, an ABox $\mathcal{A}$, and a Boolean query q, we define:

- *P-Cens-Ent* as the problem of deciding whether $\mathcal{T} \cup cens(\mathcal{A}) \models q$ for each $cens(\cdot) \in PCens(\mathcal{E}_{\succ})$;
- *G-Cens-Ent* as the problem of deciding whether $\mathcal{T} \cup cens(\mathcal{A}) \models q$ for each $cens(\cdot) \in GCens(\mathcal{E}_{\succ})$;
- *IP-Cens-Ent* as the problem of deciding whether $\mathcal{T} \cup cens_{\mathsf{IP}}(\mathcal{A}) \models q$, where $cens_{\mathsf{IP}}(\cdot)$ is the IP censor for $\mathcal{E}_{\succ}$;
- *IG-Cens-Ent* as the problem of deciding whether $\mathcal{T} \cup cens_{\mathsf{IG}}(\mathcal{A}) \models q$, where $cens_{\mathsf{IG}}(\cdot)$ is the IG censor for $\mathcal{E}_{\succ}$.

The following relationships between the above problems are immediate to verify:

- for every input for which P-Cens-Ent answers true, also G-Cens-Ent does;
- for every input for which IP-Cens-Ent answers true, also P-Cens-Ent does;
- for every input for which IG-Cens-Ent answers true, also G-Cens-Ent does.

From the outcomes presented in [25], which already hold for BCQs, it immediately derives what follows.

Corollary 2. *For prioritized DL-Lite$_{\mathcal{R}}$ CQE specifications and BCQs:*

- *P-Cens-Ent and IP-Cens-Ent are coNP-hard in data complexity;*
- *G-Cens-Ent and IG-Cens-Ent are Π_2^p-hard in data complexity.*

Results in Corollary 2 represent a clear obstacle to the use of the above forms of priority-based censors over real-world, large datasets. In the next section, we see how these censors can be suitably approximated for practical use.

4.2 Deterministic priority-based censors

In this section, we first give a deterministic notion of priority-based censor (DD censor), which is an adaptation in our framework of the one studied in [25] in the context of CQA, and its parameterized sound approximation called k-DD censor.

Corollary 2 clearly says that under Pareto-optimal or Globally-optimal censors, or their intersection-based versions, entailment of BCQs is inherently non-deterministic (unless P = NP). Towards the identification of a tractable approximation, we give below the notion of *deterministically disclosed atoms* (DDA) and *deterministically censored atoms* (DCA). Hereinafter, given a priority relation $\succ$, a fact α, and a set of facts $\mathcal{F}$, we write $\alpha \succ \mathcal{F}$ if there exists a fact $\beta \in \mathcal{F}$ such that $\alpha \succ \beta$. Moreover, given a fact α, we denote by $inSec(\mathcal{E}_\succ, \mathcal{A}, \alpha)$ the set of secrets $\mathcal{F} \in secrets(\mathcal{E}_\succ, \mathcal{A})$ such that $\alpha \in \mathcal{F}$.

Definition 15. [DDA and DCA] Given a prioritized CQE specification $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$ and an ABox $\mathcal{A}$, we denote by $DDA(\mathcal{E}_\succ, \mathcal{A})$ and $DCA(\mathcal{E}_\succ, \mathcal{A})$ the inclusion-minimal subsets of $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ such that:

$$DDA(\mathcal{E}_\succ, \mathcal{A}) = \{\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A}) \mid \forall \mathcal{F} \in inSec(\mathcal{E}_\succ, \mathcal{A}, \alpha)\ \exists \beta \in \mathcal{F} \text{ s.t. } \alpha \succ \beta \vee (\alpha \neq \beta \wedge \beta \in DCA(\mathcal{E}_\succ, \mathcal{A}))\}$$

$$DCA(\mathcal{E}_\succ, \mathcal{A}) = \{\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A}) \mid \exists \mathcal{F} \in inSec(\mathcal{E}_\succ, \mathcal{A}, \alpha) \text{ s.t. } \mathcal{F} \setminus DDA(\mathcal{E}_\succ, \mathcal{A}) = \{\alpha\}\}$$

In words, an atom $\alpha \in DDA$ is such that α does not occur in any secret or, either, in each secret in which it occurs there is an atom β such that $\alpha \succ \beta$ or $\beta \in DCA$. Instead, an atom in DCA is such that there is a secret where it is the only atom not in DDA. It is immediate to verify that $DDA(\mathcal{E}_\succ, \mathcal{A})$ and $DCA(\mathcal{E}_\succ, \mathcal{A})$ are unique for a given pair $(\mathcal{E}_\succ, \mathcal{A})$. We are now ready to provide the definition of *Deterministically Disclosing (DD) censor*.

Definition 16. [DD censor] Let $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$ be a prioritized CQE specification. The *Deterministically Disclosing (DD) censor* for $\mathcal{E}_\succ$ is the function $cens_{\mathsf{DD}}(\cdot)$ such that, for every ABox $\mathcal{A}$:

$$cens_{\mathsf{DD}}(\mathcal{A}) = DDA(\mathcal{E}_\succ, \mathcal{A}).$$

Example 21. Let $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$, and let $\mathcal{A}$ be as in Example 20 and Example 7, respectively. The set of deterministically disclosed atoms is:

$$cens_{\mathsf{DD}}(\mathcal{A}) = DDA(\mathcal{E}_\succ, \mathcal{A}) = \{\mathsf{WorksFor}(\mathsf{bob}, \mathsf{a}_1), \mathsf{WorksFor}(\mathsf{ann}, \mathsf{a}_2), \mathsf{Manages}(\mathsf{ann}, \mathsf{a}_2), \mathsf{BreachedBy}(\mathsf{a}_2, \mathsf{h}_2), \mathsf{GAgency}(\mathsf{a}_2)\}.$$

On the other hand, the set of deterministically censored atoms is $DCA(\mathcal{E}_\succ, \mathcal{A}) = \{\mathsf{SServ}(\mathsf{a}_2)\}$. ▽

The proposition below follows immediately from the definition of DD censor[2].

Proposition 10. *Let $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \emptyset \rangle$ be a prioritized CQE specification with an empty priority relation. The DD censor for $\mathcal{E}_\succ$ coincides with the IGA censor for $\langle \mathcal{T}, \mathcal{P} \rangle$.*

[2] A similar result is provided in [25, Theorem 38] in the context of CQA.

It is also easy to verify that the DD censor satisfies the property given in Definition 6.

Proposition 11. *Let $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$ be a prioritized CQE specification. The DD censor $cens_{\mathsf{DD}}(\cdot)$ for $\mathcal{E}_\succ$ satisfies the indistinguishability property.*

The following proposition, whose proof is straightforward, establishes the relationship between DD censors and the previously presented IP and IG censors.

Proposition 12. *Let $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$ be a prioritized CQE specification, and let $cens_{\mathsf{IP}}(\cdot)$ and $cens_{\mathsf{IG}}(\cdot)$ be the IP censor and IG censor for $\mathcal{E}_\succ$, respectively. Then, for every ABox $\mathcal{A}$ we have that:*

$$cens_{\mathsf{DD}}(\mathcal{A}) \subseteq cens_{\mathsf{IP}}(\mathcal{A}) \subseteq cens_{\mathsf{IG}}(\mathcal{A}).$$

Query entailment under DD censors is defined as usual.

Definition 17. [DD-Cens-Ent] Given as input a prioritized CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, an ABox $\mathcal{A}$, and a Boolean query q, DD-Cens-Ent is the problem of deciding whether $\mathcal{T} \cup cens_{\mathsf{DD}}(\mathcal{A}) \models q$. In case this holds, we write $\langle \mathcal{E}_\succ, \mathcal{A} \rangle \models_{\mathsf{DD}} q$.

From Proposition 12, it follows that every input for which DD-Cens-Ent answers true is also such that IP-Cens-Ent (and consequently IG-Cens-Ent) does.

Given a prioritized CQE specification $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$ and an ABox $\mathcal{A}$, it is not difficult to see that $DDA(\mathcal{E}_\succ, \mathcal{A})$ and $DCA(\mathcal{E}_\succ, \mathcal{A})$ correspond to the least fixpoint of the equations:

$$\begin{aligned} DDA_{i+1}(\mathcal{E}_\succ, \mathcal{A}) = \{\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A}) \mid \forall \mathcal{F} \in inSec(\mathcal{E}_\succ, \mathcal{A}, \alpha)\ \exists \beta \in \mathcal{F} \text{ s.t.} \\ \alpha \succ \beta \vee (\alpha \neq \beta \wedge \beta \in DCA_i(\mathcal{E}_\succ, \mathcal{A}))\} \\ DCA_{i+1}(\mathcal{E}_\succ, \mathcal{A}) = \{\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A}) \mid \exists \mathcal{F} \in inSec(\mathcal{E}_\succ, \mathcal{A}, \alpha) \text{ s.t.} \\ \mathcal{F} \setminus DDA_i(\mathcal{E}_\succ, \mathcal{A}) = \{\alpha\}\} \end{aligned}$$

where $DDA_0(\mathcal{E}_\succ, \mathcal{A}) = DCA_0(\mathcal{E}_\succ, \mathcal{A}) = \emptyset$. Notice that, by definition, $DDA_i(\mathcal{E}_\succ, \mathcal{A}) = DDA_{i+1}(\mathcal{E}_\succ, \mathcal{A})$ holds for every odd integer i. For a prioritized DL-Lite$_\mathcal{R}$ CQE specifications $\mathcal{E}_\succ$ and an ABox $\mathcal{A}$, computing such fixpoint is in P in the size of $\mathcal{A}$, and from the results in [25] it also follows that, in case of BCQs, the decision problem DD-Cens-Ent is P-hard in data complexity.

By fixing a positive integer k, we can define a new censor that we call *k-DD censor*.

Definition 18. [k-DD censor] Let $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$ be a prioritized CQE specification and k be a positive integer. The *k-DD censor* for $\mathcal{E}_\succ$ is the function $cens_{\mathsf{DD_k}}(\cdot)$ such that, for every ABox $\mathcal{A}$:

$$cens_{\mathsf{DD_k}}(\mathcal{A}) = DDA_k(\mathcal{E}_\succ, \mathcal{A}).$$

We next define Boolean query entailment under k-DD censors.

Definition 19. [kDD-Cens-Ent] Given as input a CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, an ABox $\mathcal{A}$, a positive integer k, and a Boolean query q, kDD-Cens-Ent is the problem of deciding whether $\mathcal{T} \cup cens_{\mathsf{DD_k}}(\mathcal{A}) \models q$. In case this holds, we write $\langle \mathcal{E}, \mathcal{A} \rangle \models^k_{\mathsf{DD}} q$.

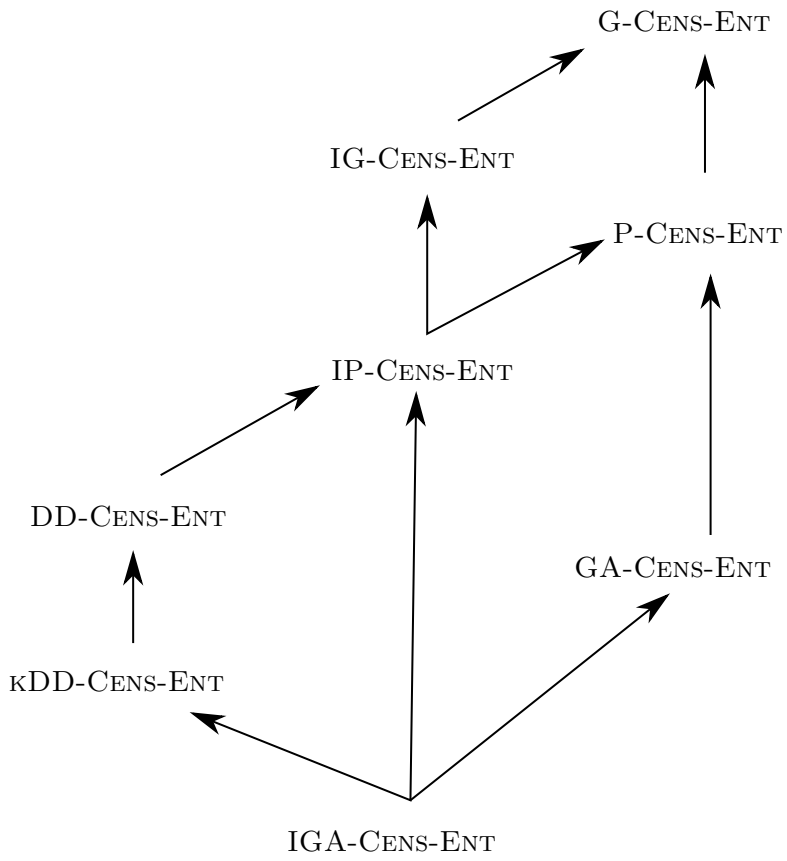

Figure 4.1. An arrow X $\longrightarrow$ Y indicates that the entailment semantics associated to problem X soundly approximates the one associated to problem Y.

Figure 4.1 summarizes all the decision problems defined until this point. In the rest of this chapter, we study the above problem for prioritized DL-Lite$_\mathcal{R}$ CQE specifications and BUCQs as queries.

Given any prioritized CQE specification $\mathcal{E}_\succ$ and any positive integer k, since $DDA_k(\mathcal{E}_\succ, \mathcal{A}) \subseteq DDA(\mathcal{E}_\succ, \mathcal{A})$ holds for every ABox $\mathcal{A}$, then the k-DD censor for $\mathcal{E}_\succ$ constitutes a sound approximation of the DD censor for $\mathcal{E}_\succ$. Thus, for every Boolean query q, $\langle \mathcal{E}_\succ, \mathcal{A}\rangle \models^k_{\mathsf{DD}} q$ implies $\langle \mathcal{E}_\succ, \mathcal{A}\rangle \models_{\mathsf{DD}} q$. Moreover, it is immediate to verify that the k-DD censor preserves the indistinguishability property.

Example 22. Let $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ\rangle$ and $\mathcal{A}$ be, respectively, the CQE specification and the ABox of the running example. For $k = 3$, the k-DD censor can be computed as follows:

$DDA_0(\mathcal{E}_\succ, \mathcal{A}) = DCA_0(\mathcal{E}_\succ, \mathcal{A}) = \emptyset$
$DDA_1(\mathcal{E}_\succ, \mathcal{A}) = \{\mathsf{WorksFor}(\mathsf{bob}, \mathsf{a}_1), \mathsf{WorksFor}(\mathsf{ann}, \mathsf{a}_2), \mathsf{Manages}(\mathsf{ann}, \mathsf{a}_2), \mathsf{GAgency}(\mathsf{a}_2)\}$
$DCA_1(\mathcal{E}_\succ, \mathcal{A}) = \emptyset$
$DDA_2(\mathcal{E}_\succ, \mathcal{A}) = DDA_1(\mathcal{E}_\succ, \mathcal{A})$
$DCA_2(\mathcal{E}_\succ, \mathcal{A}) = \{\mathsf{SServ}(\mathsf{ann})\}$
$DDA_3(\mathcal{E}_\succ, \mathcal{A}) = DDA_2(\mathcal{E}_\succ, \mathcal{A}) \cup \{\mathsf{BreachedBy}(\mathsf{a}_2, \mathsf{h}_2)\}$

Note that for $k = 3$ we reach the fixpoint, i.e. $DDA(\mathcal{E}_\succ, \mathcal{A}) = DDA_3(\mathcal{E}_\succ, \mathcal{A})$ and

$DCA(\mathcal{E}_\succ, \mathcal{A}) = DCA_3(\mathcal{E}_\succ, \mathcal{A})$. ▽

In what follows, we show that kDD-Cens-Ent is FO rewritable in our considered scenario for any positive integer k, that is, for every prioritized DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}_\succ$ and BUCQ q, one can effectively compute an FO query q_r such that, for every ABox $\mathcal{A}$, $\langle \mathcal{E}_\succ, \mathcal{A} \rangle \models^k_{\mathsf{DD}} q$ iff q_r evaluates to true in $\mathcal{A}$.

4.3 Query answering under k-DD censor

We now give our query rewriting technique for solving kDD-Cens-Ent.

Hereafter, with a slight abuse of notation, given two atoms $\alpha = S_1(\mathbf{x})$ and $\beta = S_2(\mathbf{y})$ and a priority relation $\succ$, we say that $\alpha \succ \beta$ if $S_1 \succ S_2$. Now, let α be an atom, Φ be a set of FO formulas, and $\succ$ be a priority relation. We denote by *notPreferred*$(\alpha, \Phi, \succ)$ the set of all formulas $\phi \in \Phi$ such that there does not occur in ϕ any atom β such that $\alpha \succ \beta$.

Let ϕ be the BCQ$^{\neq}$ $\exists \mathbf{x}\,(\alpha \wedge \beta_1 \wedge \ldots \wedge \beta_n \wedge \lambda_1 \wedge \ldots \wedge \lambda_h)$, where α and each β_i are predicate atoms and each λ_i is an inequality, and let $\mathcal{E}_\succ$ be a prioritized DL-Lite$_\mathcal{R}$ CQE specification. We define:

- $allDD_i(\phi, \alpha, \mathcal{E}_\succ) = \exists \mathbf{y}\,(DD_i(\beta_1, \mathcal{E}_\succ) \wedge \ldots \wedge DD_i(\beta_n, \mathcal{E}_\succ))$, where $\mathbf{y}$ are the variables in $\mathbf{x}$ that do not occur in α;
- $oneDC_i(\phi, \alpha, \mathcal{E}_\succ) = DC_i(\beta_1, \mathcal{E}_\succ) \vee \ldots \vee DC_i(\beta_n, \mathcal{E}_\succ)$.

By convention, if $i = 0$, then $allDD_0(\phi, \alpha, \mathcal{E}_\succ) = \mathit{true}$ and $oneDC_0(\phi, \alpha, \mathcal{E}_\succ) = \mathit{false}$. Moreover, we denote by $\mathcal{Q}_\mathcal{P}$ the set of BCQ$^{\neq}$s corresponding to the extended denials returned by *MinPolicy*$(\mathcal{P}, \mathcal{T})$, i.e. $\mathcal{Q}_\mathcal{P} = \{q_\delta \mid \delta \in \mathit{MinPolicy}(\mathcal{P}, \mathcal{T})\}$.

For an atom α, a prioritized DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$, and a positive integer i, we define $DD_i(\alpha, \mathcal{E}_\succ)$ as follows:

$$DD_i(\alpha, \mathcal{E}_\succ) = \alpha \wedge \left(\bigwedge_{\substack{q_\delta \in \mathit{notPreferred}(\alpha, \mathcal{Q}_\mathcal{P}, \succ), \\ \sigma \in \mathit{findSub}(\alpha, q_\delta)}} \forall \mathbf{w}\ (\neg \sigma(q_\delta) \vee \mathit{oneDC}_{i-1}(\sigma(q_\delta), \alpha, \mathcal{E}_\succ)) \right),$$

where $\mathbf{w}$ contains all the variables occurring in $\sigma(q_\delta)$ but not in α, and *findSub*$(\cdot, \cdot)$ is defined as in Section 3.3.1. In case $i = 0$, we impose $DD_0(\alpha, \mathcal{E}_\succ) = \mathit{false}$.

We also define $DC_i(\alpha, \mathcal{E}_\succ)$ as follows:

$$DC_i(\alpha, \mathcal{E}_\succ) = \bigvee_{\substack{q_\delta \in \mathit{notPreferred}(\alpha, \mathcal{Q}_\mathcal{P}, \succ), \\ \sigma \in \mathit{findSub}(\alpha, q_\delta)}} \exists \mathbf{v}\,(\sigma(q_\delta) \wedge \mathit{allDD}_{i-1}(\sigma(q_\delta), \alpha, \mathcal{E}_\succ)),$$

where $\mathbf{v}$ contains all the variables occurring in $\sigma(q_\delta)$ but not in α. In case $i = 0$, we impose $DC_0(\alpha, \mathcal{E}_\succ) = \mathit{false}$.

Given a BUCQ q, a prioritized DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}_\succ$, and a positive integer k, we define the Boolean FO query $DDQ(q, \mathcal{E}_\succ, k)$ as follows:

$$DDQ(q, \mathcal{E}_\succ, k) = \bigvee_{\exists \mathbf{x}\, \gamma \in q} \left(\exists \mathbf{x} \bigwedge_{\alpha \in \mathit{predAt}(\gamma)} DD_k(\alpha, \mathcal{E}_\succ) \right).$$

Finally, for a BUCQ q, a prioritized DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$, and a positive integer k, we define:

$$\mathit{DDRew}(q, \mathcal{E}_\succ, k) = \mathit{AtomRewr}(\mathit{DDQ}(\mathit{PerfectRef}(q, \mathcal{T}), \mathcal{E}_\succ, k), \mathcal{T}).$$

Notice that, for every odd i, $DD_i(\alpha, \mathcal{E}_\succ) = DD_{i+1}(\alpha, \mathcal{E}_\succ)$ (by definition), and thus $\mathit{DDRew}(q, \mathcal{E}_\succ, i) = \mathit{DDRew}(q, \mathcal{E}_\succ, i{+}1)$.

Example 23. Consider the prioritized DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}_\succ = \langle \mathcal{T}, \mathcal{P}, \succ \rangle$ and BUCQ q of the running example. For computing the reformulated query $\mathit{DDQ}(q, \mathcal{E}_\succ, k)$ for $k = 3$, we first need to compute the formula $DD_3(\alpha, \mathcal{E}_\succ)$ for each atom α occurring in $\mathit{PerfectRef}(q, \mathcal{T})$ (recall that $\mathit{PerfectRef}(q, \mathcal{T}) = \{\exists x, y\, \mathsf{OperatesIn}(x, y), \exists x, y\, \mathsf{BreachedBy}(x, y), \exists x\, \mathsf{SServ}(x)\}$). We proceed as follows.

$DD_0(\alpha, \mathcal{E}_\succ) = DC_0(\alpha, \mathcal{E}_\succ) = \mathit{false}$, for each atom α.

$$
\begin{aligned}
&DD_1(\mathsf{GAgency}(x), \mathcal{E}_\succ) = \mathsf{GAgency}(x)\\
&DD_1(\mathsf{Manages}(x, y), \mathcal{E}_\succ) = \mathsf{Manages}(x, y)\\
&DD_1(\mathsf{WorksFor}(x, y), \mathcal{E}_\succ) = \mathsf{WorksFor}(x, y)\\
&DD_1(\mathsf{OperatesIn}(x, y), \mathcal{E}_\succ) = \mathsf{OperatesIn}(x, y)\\
&DD_1(\mathsf{BreachedBy}(x, y), \mathcal{E}_\succ) = \mathsf{BreachedBy}(x, y) \wedge (\neg(\mathsf{BreachedBy}(x, y) \wedge \mathsf{SServ}(x)) \vee\\
&\qquad\qquad DC_0(\mathsf{SServ}(x), \mathcal{E}_\succ))\\
&\qquad\qquad \equiv \mathsf{BreachedBy}(x, y) \wedge \neg\mathsf{SServ}(x)\\
&DD_1(\mathsf{SServ}(x), \mathcal{E}_\succ) = \mathsf{SServ}(x) \wedge\\
&\qquad \forall y\, (\neg(\mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(x, y)) \vee\\
&\qquad\quad DC_0(\mathsf{BreachedBy}(x, y), \mathcal{E}_\succ)) \wedge\\
&\qquad \forall y\, (\neg(\mathsf{SServ}(x) \wedge \mathsf{Manages}(y, x)) \vee DC_0(\mathsf{Manages}(y, x), \mathcal{E}_\succ)) \wedge\\
&\qquad \forall y\, (\neg(\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y, x)) \vee DC_0(\mathsf{WorksFor}(y, x), \mathcal{E}_\succ))\\
&\qquad \equiv \mathsf{SServ}(x) \wedge \forall y\, (\neg\mathsf{BreachedBy}(x, y)) \wedge \forall y\, (\neg\mathsf{Manages}(y, x)) \wedge\\
&\qquad \forall y\, (\neg\mathsf{WorksFor}(y, x)).
\end{aligned}
$$

$DC_1(\alpha, \mathcal{E}_\succ) = \mathit{false}$, for each atom α.

$DD_2(\alpha, \mathcal{E}_\succ) = DD_1(\alpha, \mathcal{E}_\succ)$, for each atom α.

$$
\begin{aligned}
&DC_2(\mathsf{Manages}(x, y), \mathcal{E}_\succ) = \mathit{false}\\
&DC_2(\mathsf{WorksFor}(x, y), \mathcal{E}_\succ) = \mathit{false}\\
&DC_2(\mathsf{BreachedBy}(x, y), \mathcal{E}_\succ) = \mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(x, y) \wedge DD_1(\mathsf{SServ}(x), \mathcal{E}_\succ)\\
&\qquad\qquad = \mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(x, y) \wedge \mathsf{SServ}(x) \wedge\\
&\qquad\qquad\quad \forall y'\, (\neg\mathsf{BreachedBy}(x, y')) \wedge \forall y'\, (\neg\mathsf{Manages}(y', x)) \wedge\\
&\qquad\qquad\quad \forall y'\, (\neg\mathsf{WorksFor}(y', x))\\
&\qquad\qquad \equiv \mathit{false}\\
&DC_2(\mathsf{SServ}(x), \mathcal{E}_\succ) = \exists y\, (\mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(x, y) \wedge DD_1(\mathsf{BreachedBy}(x, y), \mathcal{E}_\succ)) \vee\\
&\qquad \exists y\, (\mathsf{SServ}(x) \wedge \mathsf{Manages}(y, x) \wedge DD_1(\mathsf{Manages}(y, x), \mathcal{E}_\succ)) \vee\\
&\qquad \exists y\, (\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y, x) \wedge DD_1(\mathsf{WorksFor}(y, x), \mathcal{E}_\succ))\\
&\qquad \equiv \exists y\, (\mathsf{SServ}(x) \wedge \mathsf{Manages}(y, x)) \vee \exists y\, (\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y, x))
\end{aligned}
$$

$DD_3(\mathsf{OperatesIn}(x, y), \mathcal{E}_\succ) = \mathsf{OperatesIn}(x, y)$

$$
\begin{aligned}
DD_3(\mathsf{BreachedBy}(x,y),\mathcal{E}_\succ) &= \mathsf{BreachedBy}(x,y) \wedge (\neg(\mathsf{BreachedBy}(x,y) \wedge \mathsf{SServ}(x)) \vee \\
&\quad DC_2(\mathsf{SServ}(x),\mathcal{E}_\succ)) \\
&\equiv \mathsf{BreachedBy}(x,y) \wedge (\neg\mathsf{SServ}(x) \vee \exists y'\,(\mathsf{SServ}(x) \wedge \\
&\quad \mathsf{Manages}(y',x)) \vee \exists y'\,(\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y',x))) \\
DD_3(\mathsf{SServ}(x),\mathcal{E}_\succ) &= \mathsf{SServ}(x) \wedge \\
&\quad \forall y\,(\neg(\mathsf{SServ}(x) \wedge \mathsf{BreachedBy}(x,y)) \vee DC_2(\mathsf{BreachedBy}(x,y),\mathcal{E}_\succ)) \wedge \\
&\quad \forall y\,(\neg(\mathsf{SServ}(x) \wedge \mathsf{Manages}(y,x)) \vee DC_2(\mathsf{Manages}(y,x),\mathcal{E}_\succ)) \wedge \\
&\quad \forall y\,(\neg(\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y,x)) \vee DC_2(\mathsf{WorksFor}(y,x),\mathcal{E}_\succ)) \\
&\equiv \mathsf{SServ}(x) \wedge \forall y\,(\neg\mathsf{BreachedBy}(x,y)) \wedge \forall y\,(\neg\mathsf{Manages}(y,x)) \wedge \\
&\quad \forall y\,(\neg\mathsf{WorksFor}(y,x)).
\end{aligned}
$$

Now, called q_r the query returned by $\mathit{PerfectRef}(q,\mathcal{T})$, we have that:

$$
\begin{aligned}
\mathit{DDQ}(q_r,\mathcal{E}_\succ,k) = {} & \exists x,y\,\mathsf{OperatesIn}(x,y) \vee \\
& \exists x\,\Big(\mathsf{SServ}(x) \wedge \forall y\,(\neg\mathsf{BreachedBy}(x,y)) \wedge \forall y\,(\neg\mathsf{Manages}(y,x)) \wedge \\
& \quad \forall y\,(\neg\mathsf{WorksFor}(y,x))\Big) \vee \\
& \exists x,y\,\Big(\Big(\neg\mathsf{SServ}(x) \vee \exists y'\,(\mathsf{SServ}(x) \wedge \mathsf{Manages}(y',x)) \vee \\
& \quad \exists y'\,(\mathsf{SServ}(x) \wedge \mathsf{WorksFor}(y',x))\Big) \wedge \mathsf{BreachedBy}(x,y)\Big),
\end{aligned}
$$

and finally:

$$
\begin{aligned}
\mathit{DDRew}(q,\mathcal{E}_\succ,k) = {} & \exists x,y\,\mathsf{OperatesIn}(x,y) \vee \\
& \exists x\,\Big(\mathsf{SServ}(x) \wedge \forall y\,(\neg\mathsf{BreachedBy}(x,y)) \wedge \forall y\,(\neg\mathsf{Manages}(y,x)) \wedge \\
& \quad \forall y\,(\neg(\mathsf{Manages}(y,x) \vee \mathsf{WorksFor}(y,x)))\Big) \vee \\
& \exists x,y\,\Big(\Big(\neg\mathsf{SServ}(x) \vee \exists y'\,(\mathsf{SServ}(x) \wedge \mathsf{Manages}(y',x)) \vee \\
& \quad \exists y'\,(\mathsf{SServ}(x) \wedge (\mathsf{Manages}(y',x) \vee \mathsf{WorksFor}(y',x)))\Big) \wedge \\
& \quad \mathsf{BreachedBy}(x,y)\Big).
\end{aligned}
$$

▽

It is easy to see that $\mathit{DDRew}(q,\mathcal{E}_\succ,k)$ is a FO sentence. The following theorem states that, for prioritized DL-Lite$_\mathcal{R}$ CQE specifications $\mathcal{E}_\succ$ and BUCQs q, the problem kDD-Cens-Ent can be reduced to the evaluation of $\mathit{DDRew}(q,\mathcal{E}_\succ,k)$ in $\mathcal{A}$, and this holds for every fixed positive integer k. In other terms, the problem is FO rewritable. As done for Theorem 2, we do this on the basis of Proposition 1 and Lemma 5.

Theorem 3. *Let $\mathcal{E}_\succ = \langle\mathcal{T},\mathcal{P},\succ\rangle$ be a prioritized DL-Lite$_\mathcal{R}$ CQE specification, let k be a positive integer, let $\mathit{cens}_{\mathsf{DD_k}}(\cdot)$ be the k-DD censor for $\mathcal{E}_\succ$, and let q be a BUCQ. For every ABox $\mathcal{A}$, we have that $\mathcal{T} \cup \mathit{cens}_{\mathsf{DD_k}}(\mathcal{A}) \models q$ iff $\mathit{DDRew}(q,\mathcal{E}_\succ,k)$ evaluates to true in $\mathcal{A}$.*

Proof. By exploiting Proposition 1 and Lemma 5, to prove the thesis of the theorem note that it is enough to show the following: given a BUCQ q, we have that q evaluates to true in $\mathit{cens}_{\mathsf{DD_k}}(\mathcal{A})$ iff $\mathit{DDQ}(q,\mathcal{E}_\succ,k)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$.

First, we prove inductively the following property: for every integer i such that $0 \leq i \leq k$, and for every ground atom α, we have that $\alpha \in \mathit{DDA}_i(\mathcal{E}_\succ,\mathcal{A})$ iff

$DD_i(\alpha, \mathcal{E}_\succ)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ and $\alpha \in DCA_i(\mathcal{E}_\succ, \mathcal{A})$ iff $DC_i(\alpha, \mathcal{E}_\succ)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$.

The base case trivially holds since $DDA_0(\mathcal{E}_\succ, \mathcal{A}) = DCA_0(\mathcal{E}_\succ, \mathcal{A}) = \emptyset$ and $DD_0(\alpha, \mathcal{E}_\succ) = DC_0(\alpha, \mathcal{E}_\succ) = \mathit{false}$.

We now prove the inductive case. We only prove that $\alpha \in DDA_i(\mathcal{E}_\succ, \mathcal{A})$ iff $DD_i(\alpha, \mathcal{E}_\succ)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ is true. One can prove in an analogous way that $\alpha \in DCA_i(\mathcal{E}_\succ, \mathcal{A})$ iff $DC_i(\alpha, \mathcal{E}_\succ)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$.

($\Rightarrow$:) Suppose that $\alpha \in DDA_i(\mathcal{E}_\succ, \mathcal{A})$, i.e. $\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ and for each secret $\mathcal{F} \in \mathit{inSec}(\mathcal{E}_\succ, \mathcal{A}, \alpha)$ either $\alpha \succ \beta$ for some $\beta \in \mathcal{F}$ or $\beta \in DCA_{i-1}(\mathcal{E}_\succ, \mathcal{A})$ for some $\beta \in \mathcal{F}$ with $\alpha \neq \beta$. Consider the formula $DD_i(\alpha, \mathcal{E}_\succ)$, which is of the form $\alpha \wedge \bigwedge \forall \mathbf{w}\,(\neg\sigma(q_\delta) \vee \mathit{oneDC}_{i-1}(\sigma(q_\delta), \alpha, \mathcal{E}_\succ))$. In particular, consider each conjunct $\forall \mathbf{w}\,(\neg\sigma(q_\delta) \vee \mathit{oneDC}_{i-1}(\sigma(q_\delta), \alpha, \mathcal{E}_\succ))$ in $DD_i(\alpha, \mathcal{E}_\succ)$. By Proposition 9, either $\sigma(q_\delta)$ evaluates to false in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, and so α does not belong to any secret that violates δ, or there is a secret $\mathcal{F}$ violating δ such that $\alpha \in \mathcal{F}$. In the former case, we are done. In the latter case, by construction of $DD_i(\alpha, \mathcal{E}_\succ)$ (in particular, the fact that $q_\delta \in \mathit{notPreferred}(\alpha, \mathcal{Q}_\mathcal{P}, \succ)$), we have that there is no $\beta \in \mathcal{F}$ with $\alpha \succ \beta$. By the assumption that $\alpha \in DDA_i(\mathcal{E}_\succ, \mathcal{A})$, it follows that $\beta \in DCA_{i-1}(\mathcal{E}_\succ, \mathcal{A})$ for some $\beta \in \mathcal{F}$ with $\alpha \neq \beta$. But then, due to the inductive hypothesis, we derive that $DC_{i-1}(\beta, \mathcal{E}_\succ)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, and therefore $\mathit{oneDC}_{i-1}(\sigma(q_\delta), \alpha, \mathcal{E}_\succ))$ is true as well. Thus, also in the latter case we have that $\forall \mathbf{w}\,(\neg\sigma(q_\delta) \vee \mathit{oneDC}_{i-1}(\sigma(q_\delta), \alpha, \mathcal{E}_\succ))$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, from which we derive that $DD_i(\alpha, \mathcal{E}_\succ)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$.

($\Leftarrow$:) Now, suppose that $DD_i(\alpha, \mathcal{E}_\succ)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, i.e. $\alpha \in cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ and each $\forall \mathbf{w}\,(\neg\sigma(q_\delta) \vee \mathit{oneDC}_{i-1}(\sigma(q_\delta), \alpha, \mathcal{E}_\succ))$ in $DD_i(\alpha, \mathcal{E}_\succ)$ is such that $\forall \mathbf{w}\,(\neg\sigma(q_\delta) \vee \mathit{oneDC}_{i-1}(\sigma(q_\delta), \alpha, \mathcal{E}_\succ))$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. From the latter, by Proposition 9, we derive that every secret $\mathcal{F}$ with $\alpha \in \mathcal{F}$ for which there is no atom $\beta \in \mathcal{F}$ with $\alpha \succ \beta$ is such that there exists an atom $\beta \in \mathcal{F}$ with $DC_{i-1}(\beta, \mathcal{E}_\succ)$ evaluating to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. By the inductive hypothesis, we derive that $\beta \in DCA_{i-1}(\mathcal{E}_\succ, \mathcal{A})$. Since this holds for every possible secret $\mathcal{F}$ with $\alpha \in \mathcal{F}$ and for which there is no atom $\beta \in \mathcal{F}$ with $\alpha \succ \beta$, by definition we have that $\alpha \in DDA_i(\mathcal{E}_\succ, \mathcal{A})$.

In light of this, one can prove the thesis of the theorem following similar considerations as done in the proof of Theorem 2 (it is indeed sufficient to replace $\mathit{Discl}(\alpha, \mathcal{E})$ with $DD_k(\alpha, \mathcal{E}_\succ)$ and $\mathit{cens}_{\mathsf{IGA}}(\mathcal{A})$ with $\mathit{cens}_{\mathsf{DD_k}}(\mathcal{A})$, and follow the same line of reasoning). ∎

Example 24. Let $\mathcal{E}_\succ$, $\mathcal{A}$ and q be as in Example 20, Example 4 and Example 11, respectively, and consider the perfect reformulation $\mathit{DDRew}(q, \mathcal{E}_\succ, 3)$ of Example 23. One can verify that $\mathit{DDRew}(q_r, \mathcal{E}_\succ, 3)$ evaluates to true in $\mathcal{A}$. Indeed, for $k = 3$ we have that $\mathcal{T} \cup \mathit{cens}_{\mathsf{DD_k}}(\mathcal{A}) \models q$.

It is also worth noting that $\mathit{DDRew}(q_r, \mathcal{E}_\succ, 1)$ evaluates to false in $\mathcal{A}$ and $\mathcal{T} \cup \mathit{cens}_{\mathsf{DD_k}}(\mathcal{A}) \not\models q$. ▽

The corollary below follows from Theorem 3 and the fact that evaluating an FO query over an ABox is in AC^0 in the size of the ABox.

Corollary 3. *kDD-Cens-Ent for prioritized DL-Lite$_\mathcal{R}$ CQE specifications and BUCQs is in AC^0 in data complexity.*

Chapter 5

Extending ontology and policy languages

In this chapter, we propose two further possible extensions of what examined so far. In particular, in Section 5.1 we show that it is possible to enhance at the same time the ontology language to DL-Lite$_\mathcal{A}$ and the policy language to denials with comparison atoms without affecting our main complexity results. Afterwards, in Section 5.2 we employ DL-Lite$^\mathcal{H}_{horn}$ as ontology language and we let the policy also contain numerical restriction axioms.

5.1 DL-Lite$_\mathcal{A}$ ontologies and policies with comparison atoms

DL-Lite$_\mathcal{A}$ is a logic belonging to the DL-Lite family. It extends DL-Lite$_\mathcal{R}$ with the possibility of using attributes and specifying functional roles and functional attributes. Basic concepts and roles are here redefined as follows:

$$B ::= A \mid \exists R \mid \exists U \qquad R ::= P \mid P^-$$

where $U \in \Gamma_A$ and the rest aligns with DL-Lite$_\mathcal{R}$. DL-Lite$_\mathcal{A}$ TBox assertions assume either the form of a DL-Lite$_\mathcal{R}$ TBox assertion or one of the following:

$$U_1 \sqsubseteq U_2 \qquad U_1 \sqsubseteq \neg U_2 \qquad \rho(U) \sqsubseteq F \qquad (\mathsf{funct}\ R) \qquad (\mathsf{funct}\ U)$$

where $\rho(U)$ denotes the range of an attribute U, i.e. the set of values to which U relates some object, and F can be one among $T_1, \ldots, T_n$, which denote unbounded pairwise-disjoint predefined *value-domains* (i.e. subsets of Γ_V), or the symbol $\top_D$, which denotes the universal value-domain coinciding with Γ_V[1]. Such assertions specify, from left to right, positive and negative inclusions between attributes, attribute *typings*, and *functionalities* of roles and attributes. A DL-Lite$_\mathcal{A}$ TBox is a finite set of assertions of the above kind, such that each basic role R or attribute U that is functional in $\mathcal{T}$ (i.e. occurs in a functionality in $\mathcal{T}$) is never specialized, i.e.

[1] As usual for DL-Lite$_\mathcal{A}$, we assume that the sets $T_1, \ldots, T_n$ are unbounded and pairwise disjoint, and that $\Gamma_V \setminus \bigcup_{i=1}^{n} T_i$ is unbounded [82].

it does not occur in the right-hand side of any positive inclusion between roles or attributes.

The semantics of a DL-Lite$_\mathcal{A}$ ontology is given in terms of interpretations that agree on the interpretation of each predefined value-domain T_i and of each value in Γ_V. More precisely, an *interpretation* $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ for a DL-Lite$_\mathcal{A}$ ontology consists of an interpretation domain $\Delta^\mathcal{I} = \Delta^\mathcal{I}_O \cup \Gamma_V$, where $\Delta^\mathcal{I}_O$ is a non-empty set, disjoint from Γ_V, called the *domain of objects*, and an interpretation function $\cdot^\mathcal{I}$. Such notion of interpretation complies with the one for DL-Lite$_\mathcal{R}$, as the function $\cdot^\mathcal{I}$ assigns an element $o^\mathcal{I}$ of $\Delta^\mathcal{I}_O$ to each individual $o \in \Gamma_I$, a subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}_O$ to each atomic concept A, and a subset $R^\mathcal{I}$ of $\Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O$ to each role R. Moreover, $\mathcal{I}$ interprets each value $v \in \Gamma_V$ with itself, and it satisfies the following equations:

$$\begin{aligned} U^\mathcal{I} &\subseteq \Delta^\mathcal{I}_O \times \Gamma_V, && \text{if } U \in \Gamma_A \\ T_i^\mathcal{I} &= T_i, \\ \top_D^\mathcal{I} &= \Gamma_V, \\ (\exists U)^\mathcal{I} &= \{o \mid \exists v\, (o, v) \in U^\mathcal{I}\}, && \text{if } U \in \Gamma_A \\ (\rho(U))^\mathcal{I} &= \{v \mid \exists o\, (o, v) \in U^\mathcal{I}\}, && \text{if } U \in \Gamma_A \end{aligned}$$

Furthermore, the notion of *satisfaction* of a DL-Lite$_\mathcal{A}$-specific TBox assertion is defined as follows:

$$\begin{aligned} \mathcal{I} &\models U_1 \sqsubseteq U_2 && \text{iff} && U_1^\mathcal{I} \subseteq U_2^\mathcal{I} \\ \mathcal{I} &\models U_1 \sqsubseteq \neg U_2 && \text{iff} && U_1^\mathcal{I} \cap U_2^\mathcal{I} = \emptyset \\ \mathcal{I} &\models (\mathsf{funct}\ R) && \text{iff} && \forall o_1, o_2, o_3\, ((o_1, o_2) \in R^\mathcal{I} \wedge (o_1, o_3) \in R^\mathcal{I} \Longrightarrow o_2 = o_3) \\ \mathcal{I} &\models (\mathsf{funct}\ U) && \text{iff} && \forall o, v_1, v_2\ \ ((o, v_1) \in U^\mathcal{I} \wedge (o, v_2) \in U^\mathcal{I} \Longrightarrow v_1 = v_2) \end{aligned}$$

Finally, for ABox assertions we have that $\mathcal{I} \models U(a, v)$ if $(a^\mathcal{I}, v) \in U^\mathcal{I}$, for every $U \in \Gamma_A$, $a \in \Gamma_I$ and $v \in \Gamma_V$. For concept and role assertions the same applies as in DL-Lite$_\mathcal{R}$.

For what concerns the policy language, we consider denials whose body possibly contains comparison atoms. Given one such denial, we indicate as *compAt*(δ) the set of comparison atoms occurring in the premise of δ.

We then provide a practical syntactic condition that we exploit in order to obtain our main computational results. We say that a denial with comparison atoms δ is *safe* with respect to a DL-Lite$_\mathcal{A}$ TBox $\mathcal{T}$ if every variable x occurring in *compAt*(δ) occurs in *predAt*(δ) only in safe attribute range positions, i.e. in atoms of the form $U(t, x)$ such that U is an attribute and there exists no basic concept $B \neq \exists U$ such that $\mathcal{T} \models B \sqsubseteq \exists U$. Then, a DL-Lite$_\mathcal{A}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ is *safe* if $\mathcal{P}$ contains only denials that are safe with respect to $\mathcal{T}$.

Example 25. Consider the scenario described in Example 1. Let $\mathcal{P}$ be a policy containing only the denial illustrated in such example. Moreover, the intensional part of the ontology is modeled through the following DL-Lite$_\mathcal{A}$ TBox:

$$\begin{aligned} \mathcal{T} = \{\ & \exists \mathsf{buy}^- \sqsubseteq \mathsf{Product}, && \exists \mathsf{age} \sqsubseteq \mathsf{Person}, \\ & \exists \mathsf{buy} \sqsubseteq \mathsf{Person}, && \rho(\mathsf{age}) \sqsubseteq \texttt{xsd:int}, \\ & \mathsf{Medicine} \sqsubseteq \mathsf{Product}, && (\mathsf{funct}\ \mathsf{age}), \\ & \mathsf{Antidysuric} \sqsubseteq \mathsf{Medicine}\ \}. \end{aligned}$$

Such a TBox sanctions that people (Person) can buy products (Product), and that antidysuric medicines (Antidysuric) are a type of medicines (Medicine), which in turn are products. Finally, age is a functional attribute ranging over integers (`xsd:int`) and whose domain is restricted to the Person concept.

Observe that the domain of age does not specialize in any basic concept. Therefore, $\langle \mathcal{T}, \mathcal{P} \rangle$ is a safe DL-Lite$_\mathcal{A}$ CQE specification. ▽

Under the safeness assumption, we can broaden the main complexity results provided until this point to the case of DL-Lite$_\mathcal{A}$ TBoxes and policies consisting of denials with comparison atoms.

First, as shown in [100], we remark that Proposition 1 also holds for DL-Lite$_\mathcal{A}$ TBoxes. It is indeed not hard to adapt the algorithm *PerfectRef* to also handle TBox assertions with attributes by treating them in a similar way to roles. For what concerns functionalities we observe that, similarly to disjointness axioms, they do not play any role in the rewriting process. Moreover, the algorithm *MinUnsatQuery* presented in [100] exploits an extension of *PerfectRef*, called *PerfectRefIdC* [42], that also supports (limited forms of) inequalities in the input query. More precisely, the rewriting steps in *PerfectRefIdC* are exactly as in *PerfectRef*, except that inequalities are treated as atomic roles, and inequality atoms are never rewritten during the rewriting process (this distinction arises from the absence of assertions involving the inequality predicate in the TBox). Moreover, *PerfectRefIdC* does not unify query atoms if the variables involved in the unification occur also in inequality atoms. One can easily further extend this algorithm to make it also work with other kinds of comparison atoms.

We show that the upper bounds for IGA-Cens-Ent (Corollary 1) and kDD-Cens-Ent (Corollary 3), still hold in this new scenario.

Theorem 4. *IGA-Cens-Ent for safe DL-Lite$_\mathcal{A}$ CQE specifications and BUCQs is in AC^0 in data complexity.*

Proof. Note that Proposition 1 and Proposition 9 can be transposed to the considered setting, as discussed. Lemma 5 remains valid as well, by extending the *AtomRewr*$(\cdot,\cdot)$ function to also rewrite the input atoms with respect to DL-Lite$_\mathcal{A}$-specific TBox assertions:

$$AtomRewr(\alpha, \mathcal{T}) = \begin{cases} \bigvee_{\mathcal{T} \models D \sqsubseteq C} D(t) \vee \\ \bigvee_{\mathcal{T} \models \exists R \sqsubseteq C} (\exists x\, R(t,x)) \vee \\ \bigvee_{\mathcal{T} \models \exists R^- \sqsubseteq C} (\exists x\, R(x,t)) \vee \\ \bigvee_{\mathcal{T} \models \exists U \sqsubseteq C} (\exists x\, U(t,x)), & \text{if } \alpha = C(t) \\ \bigvee_{\mathcal{T} \models S \sqsubseteq R} S(t_1,t_2) \vee \\ \bigvee_{\mathcal{T} \models S^- \sqsubseteq R} S(t_2,t_1), & \text{if } \alpha = R(t_1,t_2) \\ \bigvee_{\mathcal{T} \models W \sqsubseteq U} W(t_1,t_2), & \text{if } \alpha = U(t_1,t_2) \end{cases}$$

where $C, D \in \Gamma_C$, $R, S \in \Gamma_R$ and $U \in \Gamma_A$.

Thus, the validity of the thesis can be demonstrated as done in the proof of Theorem 2. ∎

Theorem 5. *kDD-Cens-Ent for safe prioritized DL-Lite$_\mathcal{A}$ CQE specifications and BUCQs is in AC^0 in data complexity.*

Proof. As in the proof of Theorem 4, we can take advantage of the fact that, with appropriate adjustments, Proposition 1, Proposition 9 and Lemma 5 remain applicable in the given scenario. Then, one can proceed as in the proof of Theorem 3. ∎

5.2 DL-Lite$^\mathcal{H}_{horn}$ ontologies and policies with number restrictions

We now introduce a further member of the DL-Lite family, called DL-Lite$^\mathcal{H}_{horn}$ [9] (also known as DL-Lite$_{R,\sqcap}$ [46]). The language for DL-Lite$^\mathcal{H}_{horn}$ concepts and roles is defined as for DL-Lite$_\mathcal{R}$ except for basic concepts, which include the special predicate $\bot$.

DL-Lite$^\mathcal{H}_{horn}$ extends DL-Lite$_\mathcal{R}$ with the possibility of specifying TBox assertions of the following form:

$$B_1 \sqcap \ldots \sqcap B_n \sqsubseteq B$$

where B and each B_i (with $1 \leq i \leq n$) are basic concepts.

With respect to the semantics of DL-Lite$_\mathcal{R}$, the definition of *satisfaction* of a DL-Lite$^\mathcal{H}_{horn}$ TBox assertion specializes as follows:

$$\mathcal{I} \models B_1 \sqcap \ldots \sqcap B_n \sqsubseteq B \quad \text{iff} \quad B_1^\mathcal{I} \cap \ldots \cap B_n^\mathcal{I} \subseteq B^\mathcal{I}$$

We also enrich the policy language of denials with (a particular form of) numerical restriction axioms [9]. Specifically, we permit axioms containing number restrictions of the "at least" type. More formally, here *numerical restrictions* are axioms of the form:

$$A_1 \sqcap \cdots \sqcap A_k \sqcap (\geq n_1 R_1) \sqcap \cdots \sqcap (\geq n_h R_h) \sqsubseteq \bot \tag{5.1}$$

where $k \geq 0$, $h \geq 0$, $k + h \geq 1$, each A_i is an atomic concept, and each $(\geq n_i R_i)$ denotes a *number restriction*, where R_i is basic role and n_i is a positive integer. We remark that each $(\geq n_i R_i)$ construct in (5.1) corresponds to the FO sentence

$$\exists x, y_1, \ldots, y_{n_i} \, (R_i(x, y_1) \wedge \ldots \wedge R_i(x, y_{n_i}) \wedge \bigwedge_{\substack{\ell, j \in \{1,\ldots,n_i\}, \\ j \neq \ell}} y_j \neq y_\ell),$$

where each atom $R(x, y_i)$ stands for $P(x, y_i)$ if R is an atomic role $P \in \Sigma_R$, or for $P(y_i, x)$ otherwise (i.e. R is the inverse P^- of an atomic role $P \in \Sigma_R$), In more intuitive terms, $(\geq n_i R_i)$ denotes the set of objects that participate at least n_i times in the role R_i. Observe that $(\geq 1R)$ is equivalent to $\exists R$.

In this section, we consider a policy $\mathcal{P}$ over a TBox $\mathcal{T}$ as a finite set of denials and numerical restriction axioms, both specified over the signature of $\mathcal{T}$.

Example 26. During the initial phase of the Covid-19 vaccination campaign, the minors who have been given both doses of the vaccine are only those with specific diseases. This sensitive information can be kept secret by the following DL concept inclusion axiom (where vaccinatedWith is the role representing that an individual has been vaccinated with a specific vaccine lot):

$$\mathsf{Minor} \sqcap (\geq 2\ \mathsf{vaccinatedWith}) \sqsubseteq \bot$$

which corresponds to the following denial with inequalities:

$$\exists p, l_1, l_2\ \mathsf{Minor}(p) \wedge \mathsf{vaccinatedWith}(p, l_1) \wedge \mathsf{vaccinatedWith}(p, l_2) \wedge l_1 \neq l_2 \rightarrow \bot$$

Now, let $\mathcal{T} = \{\mathsf{Minor} \sqsubseteq \mathsf{Person}, \exists \mathsf{vaccinatedWith}^- \sqsubseteq \mathsf{vaccineLot}\}$, i.e. $\mathcal{T}$ says that minors are persons and that an individual can be vaccinated only with vaccine lots, and consider the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$. The function $cens(\cdot)$ below is an optimal GA censor for $\mathcal{E}$:

cens$(\cdot)$**:** given an ABox $\mathcal{A}$, $cens(\mathcal{A})$ returns the set of ground atoms obtained by removing $\mathsf{Minor}(o)$ from $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, for each individual o such that, for some individuals l_1 and l_2, both $\mathsf{vaccinatedWith}(o, l_1)$ and $\mathsf{vaccinatedWith}(o, l_2)$ are in $\mathcal{A}$.

Consider now the following ABox for $\mathcal{E}$:

$$\mathcal{A} = \{\mathsf{Minor}(\mathsf{sam}), \mathsf{vaccinatedWith}(\mathsf{sam}, \mathsf{l21}), \mathsf{vaccinatedWith}(\mathsf{sam}, \mathsf{l85}), \mathsf{Minor}(\mathsf{tom}), \mathsf{vaccinatedWith}(\mathsf{tom}, \mathsf{l44})\ \}.$$

We have that:

$$cens_{\mathsf{IGA}}(\mathcal{A}) = \{\mathsf{Person}(\mathsf{sam}), \mathsf{Person}(\mathsf{tom}), \mathsf{Minor}(\mathsf{tom}), \mathsf{vaccinatedWith}(\mathsf{tom}, \mathsf{l44}), \mathsf{vaccineLot}(\mathsf{l21}), \mathsf{vaccineLot}(\mathsf{l85}), \mathsf{vaccineLot}(\mathsf{l44})\ \}. \quad \triangledown$$

We now study the data complexity of IGA-Cens-Ent for CQE specifications whose TBox is specified in DL-Lite$^{\mathcal{H}}_{horn}$. We first notice that simply combining a TBox in this language with a policy containing numerical restriction axioms leads to dealing with ontologies where standard BCQ entailment does not enjoy the favorable computational property of the logics of the DL-Lite family, i.e. this problem is not in AC^0 in data complexity [9, 46]. To regain membership in AC^0, DL-Lite dialects allowing for both role inclusions and number restrictions impose a syntactic condition that limits the interaction between these two constructs. To this aim, we here cast in our framework condition (A$_3$) adopted in [9]. Namely, we say that a theory $\mathfrak{T}$ consisting of DL-Lite$^{\mathcal{H}}_{horn}$ TBox assertions, denials, and numerical restriction axioms is *safe* if, for every number restriction $(\geq n_i R_i)$, with $n_i \geq 2$, occurring in an axiom of the form (5.1) in $\mathfrak{T}$ we have that both R_i and R_i^- do not occur in the right-hand side of role inclusion assertions in $\mathfrak{T}$. Then, we say that a DL-Lite$^{\mathcal{H}}_{horn}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ is *safe* if $\mathcal{T} \cup \mathcal{P}$ is safe.

Under the above assumption, numerical restriction axioms fall within the axioms expressible in DL-Lite$^{(\mathcal{HN})}_{horn}$ (which DL-Lite$^{\mathcal{H}}_{horn}$ is a fragment of), for which BUCQ entailment is in AC^0 [9, Theorem 7.1]. In this case, following analogous considerations as in Section 5.1, the algorithm *PerfectRef* can be adapted in such a way

that Proposition 1 holds also when the ontology language is DL-Lite$^{\mathcal{H}}_{horn}$ and the query to be rewritten is the body of a numerical restriction axiom. We also recall, as outlined for the case of denials with comparison atoms, that *MinPolicy* supports the presence of inequalities (note that numerical restriction axioms are a particular form of denial with inequalities), which guarantees Proposition 9 to remain valid.

Theorem 6. *IGA-Cens-Ent for safe DL-Lite$^{\mathcal{H}}_{horn}$ CQE specifications and BUCQs is in AC^0 in data complexity.*

Proof (sketch). We proceed as in the proof of Theorem 4. First, note that Proposition 1 and Proposition 9 can be transposed to the considered setting, as discussed. Lemma 5 remains valid as well, by extending the function $\mathit{AtomRewr}(\cdot,\cdot)$ to also rewrite the atoms of a formula with respect DL-Lite$^{\mathcal{H}}_{horn}$ TBox axioms. Formally, given a DL-Lite$^{\mathcal{H}}_{horn}$ TBox and a BCQ q, $\mathit{AtomRewr}(q,\mathcal{T})$ substitutes each atom α of q with the formula $\mathit{AtomRewr}(\alpha,\mathcal{T})$ defined as follows:

$$\mathit{AtomRewr}(A(t),\mathcal{T}) = \bigvee_{\mathcal{T}\models\prod_{k=1}^{n} B_k\sqsubseteq A} \bigwedge_{k=1}^{n} \alpha_k(t);$$

$$\mathit{AtomRewr}(R(t_1,t_2),\mathcal{T}) = \bigvee_{\mathcal{T}\models S\sqsubseteq R} S(t_1,t_2) \quad \vee \bigvee_{\mathcal{T}\models S^-\sqsubseteq R} S(t_2,t_1)$$

where A is an atomic concept, each B_k is a basic concept, R and S are basic roles, t, t_1 and t_2 are terms and, for each $1 \leq k \leq n$, the conjunct $\alpha_k(t)$ can assume one of the following forms:

- $\alpha_k(t) = B_k(t)$, if B_k is an atomic concept;
- $\alpha_k(t) = \exists x\, P(t,x)$, if $B_k = \exists P$, where P is an atomic role;
- $\alpha_k(t) = \exists x\, P(x,t)$, if $B_k = \exists P^-$, where P is an atomic role.

Thus, the proof can proceed analogously to the one of Theorem 2. ∎

Theorem 7. *kDD-Cens-Ent for safe prioritized DL-Lite$^{\mathcal{H}}_{horn}$ CQE specifications and BUCQs is in AC^0 in data complexity.*

Proof (sketch). As in the proof of Theorem 6, we can exploit the fact that, *mutatis mutandis*, Proposition 1, Proposition 9 and Lemma 5 remain valid in the given scenario. Then, one can prove FO rewritability by adopting the same approach used in the proof of Theorem 3. ∎

Chapter 6

Practical implementation and experiments

In this chapter, we present a practical way of implementing the algorithms for Controlled Query Evaluation that we presented. In particular, we focus on the scenario described in Section 5.1, i.e. the case when the ontology language is DL-Lite$_{\mathcal{A}}$ and the policy consists of denials with (a particular form of) comparison atoms.

In the first part we show an original way of solving the problem of query entailment under censors, i.e. by encoding it into a query answering problem in OBDA. We also propose an important optimization for our rewriting technique. Finally, we perform our experiment using an OBDA benchmark and discuss our results.

6.1 Policy-protected ontology-based data access

We start by introducing the formal notion of PPOBDA specification, which generalizes CQE and OBDA specifications.

Definition 20. [PPOBDA] A *policy-protected OBDA (PPOBDA) specification* is a quadruple $\langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ such that $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ is an OBDA specification, and $\langle \mathcal{T}, \mathcal{P} \rangle$ is a CQE specification.

The semantics of a PPOBDA specification $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ coincides with that of the OBDA specification $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, and thus we naturally extend to PPOBDA the notion of source database $\mathcal{D}$, retrieved ABox (denoted $ret(\mathcal{H}, \mathcal{D})$) and setting (denoted $\langle \mathcal{H}, \mathcal{D} \rangle$).

Example 27. Consider the PPOBDA specification $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$, where $\langle \mathcal{T}, \mathcal{P} \rangle$ is the safe DL-Lite$_{\mathcal{A}}$ CQE specification of Example 25, the predicates of $\mathcal{M}$ are as follows:

$$\exists a, w\,(\texttt{person}(p, a, w)) \rightsquigarrow \mathsf{Person}(p),$$
$$\exists w\,(\texttt{person}(p, a, w)) \rightsquigarrow \mathsf{age}(p, a),$$
$$\exists t\,(\texttt{transaction}(t, pe, pr)) \rightsquigarrow \mathsf{buy}(pe, pr),$$
$$\exists e, b, c\,(\texttt{medicine}(m, e, b, c) \wedge \texttt{api}(b, \texttt{`phenazopyridine'})) \rightsquigarrow \mathsf{Antidysuric}(m),$$
$$\exists e, b, c\,(\texttt{medicine}(m, e, b, c)) \rightsquigarrow \mathsf{Medicine}(m),$$

and $\mathcal{S}$ contains the predicates below:

person		
ann	67	53
bob	59	74

medicine			
m_a	2024-10	myPill	PharmaCorp
m_b	2024-12	myPill	PharmaCorp

transaction		
1	ann	m_a
2	bob	m_b

api	
myPill	'phenazopyridine'

Table 6.1. Tables contained in the source database.

- `person`, of arity 3, whose attributes indicate the person ID, along with her age and weight;
- `api`, of arity 2, whose attributes indicate the medicine brand and the name of an active pharmaceutical ingredient (API) contained in it;
- `transaction`, of arity 3, whose attributes indicate the transaction ID, a person ID and a product ID;
- `medicine`, of arity 3, whose attributes indicate a product ID, its expiration date, the brand name, and the ID of the company selling it.

Consider also the dataset instance $\mathcal{D}$ illustrated in Table 6.1. Then, the retrieved ABox $\mathit{ret}(\mathcal{H}, \mathcal{D})$ will be:

$$\begin{aligned}\mathit{ret}(\mathcal{H}, \mathcal{D}) = \{\ & \mathsf{Person}(\mathsf{ann}), \mathsf{Person}(\mathsf{bob}), \mathsf{age}(\mathsf{ann}, 67), \mathsf{age}(\mathsf{bob}, 59), \\ & \mathsf{Antidysuric}(\mathsf{m}_a), \mathsf{Antidysuric}(\mathsf{m}_b), \mathsf{Medicine}(\mathsf{m}_a), \mathsf{Medicine}(\mathsf{m}_b), \\ & \mathsf{buy}(\mathsf{ann}, \mathsf{m}_a), \mathsf{buy}(\mathsf{bob}, \mathsf{m}_b)\ \}\end{aligned}$$

$\triangledown$

The notion of censor can be adapted to this setting as follows:

Definition 21. [Censor for a PPOBDA specification] Given a PPOBDA specification $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ and a language $\mathcal{L}_c$, a *censor* for $\mathcal{H}$ in $\mathcal{L}_c$ is a function $\mathit{cens}(\cdot)$ such that, for each source database $\mathcal{D}$ for $\mathcal{H}$, returns a set $\mathit{cens}(\mathcal{D}) \subseteq \mathcal{L}_c$ such that:

(*i*) $\langle \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, \mathcal{D} \rangle \models \phi$, for each sentence $\phi \in \mathit{cens}(\mathcal{D})$;

(*ii*) $\mathcal{T} \cup \mathcal{P} \cup \mathit{cens}(\mathcal{D})$ is a consistent FO theory.

Observe that, by construction, the language $\mathcal{L}_c$ only contains sentences specified over predicates occurring in $\mathit{pred}(\mathcal{T})$. Note also that Definition 21 coincides with Definition 4 in the case every $\mathcal{D}$ is a mirrored database with respect to an ABox $\mathcal{A}$ (i.e. $\mathcal{D} = \{P'(\mathbf{c}) \mid P' \in \mathcal{S} \wedge P(\mathbf{c}) \in \mathcal{A}\}$) and $\mathcal{M}$ is the identity mapping, i.e. a set consisting of a mapping assertion of the form $P'(\mathbf{x}) \rightsquigarrow P(\mathbf{x})$ for every k-ary predicate $P \in \mathit{pred}(\mathcal{T})$, where P' is a k-ary predicate specified in the schema $\mathcal{S}$. In this sense, the new definition of censor is a generalization of the previous one.

The notion of optimality in $\mathcal{L}_c$ is given as usual. We also overload the functions $\mathit{optCQCens}(\cdot)$ and $\mathit{optGACens}(\cdot)$ by letting them take as input a PPOBDA specification $\mathcal{H}$ and return the set of all the optimal CQ censors for $\mathcal{H}$ and all the optimal GA censors for $\mathcal{H}$, respectively.

The following proposition characterizes the relationship between optimal censors for CQE specifications and optimal censors for PPOBDA specifications.

Proposition 13. *Let $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ be a PPOBDA specification, let $\mathcal{L}_c \subseteq$ **FO** be a language, and let $cens(\cdot)$ be an optimal censor for the CQE specification $\langle \mathcal{T}, \mathcal{P} \rangle$ in $\mathcal{L}_c$. Then, the function $cens'(\cdot)$ such that $cens'(\mathcal{D}) = cens(ret(\mathcal{H}, \mathcal{D}))$ for every database $\mathcal{D}$ for $\mathcal{S}$ is an optimal censor for $\mathcal{H}$ in $\mathcal{L}_c$.*

Proof. By Definition 4, a censor $cens(\cdot)$ for $\langle \mathcal{T}, \mathcal{P} \rangle$ in $\mathcal{L}_c$ is such that, for every ABox $\mathcal{A}$, (*i*) $cens(\mathcal{A}) \subseteq cl^{\mathcal{T}}_{\mathcal{L}_c}(\mathcal{A})$ and (*ii*) $\mathcal{T} \cup \mathcal{P} \cup cens(\mathcal{A})$ is a consistent FO theory. This is true in particular, given any source database $\mathcal{D}$ for $\mathcal{S}$, for the ABox $\mathcal{R} = ret(\mathcal{H}, \mathcal{D})$. Anyway, since $cens(\mathcal{R}) \subseteq cl^{\mathcal{T}}_{\mathcal{L}_c}(\mathcal{R})$, then every $\mathcal{L}_c$ sentence $\phi \in cens(\mathcal{R})$ is also such that $\mathcal{T} \cup \mathcal{R} \models \phi$, and thus $\langle \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, \mathcal{D} \rangle \models \phi$. Then, the function $cens'(\cdot)$ as defined in the main statement is a censor for $\mathcal{H}$ in $\mathcal{L}_c$.

Now, by way of contradiction, let $cens'(\cdot)$ be not optimal. In this case, there would exist a censor $cens''(\cdot)$ such that $cens''(\mathcal{D}) \supseteq cens'(\mathcal{D})$ for every database $\mathcal{D}$ and, for some $\mathcal{D}'$, there would exist some $\mathcal{L}_c$ sentence $\psi \in cens''(\mathcal{D}') \setminus cens'(\mathcal{D}')$. By Definition 21, $\psi \in cens''(\mathcal{D}')$ implies that $\langle \mathcal{H}, \mathcal{D}' \rangle \models \psi$. Let now $\mathcal{R}' = ret(\mathcal{H}, \mathcal{D}')$. Since $cens'(\mathcal{D}') = cens(\mathcal{R}')$ holds by construction, then we have that $\psi \notin cens(\mathcal{R}')$. Anyway, since $cens(\cdot)$ is optimal, then either $\mathcal{T} \cup \mathcal{R}' \not\models \psi$ or the theory $\mathcal{T} \cup \mathcal{P} \cup cens(\mathcal{R}') \cup \{\psi\}$ is inconsistent. These two cases would contradict, respectively, the fact that $\langle \mathcal{H}, \mathcal{D}' \rangle \models \psi$ and the fact that $cens''(\cdot)$ is a censor for $\mathcal{H}$ (because $cens''(\mathcal{D}') \supseteq \mathcal{R}' \cup \{\psi\}$ by construction), which proves the thesis. ∎

We now give the notation for the semantics of skeptical entailment under CQ censors and under GA censors.

Definition 22. Let $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ be a PPOBDA specification, let $\mathcal{D}$ be a database instance for $\mathcal{S}$, and let q be a BUCQ.

- we write $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{CQ}} q$ if $\mathcal{T} \cup cens(\mathcal{D}) \models q$ for every $cens(\cdot) \in optCQCens(\mathcal{H})$;
- we write $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{GA}} q$ if $\mathcal{T} \cup cens(\mathcal{D}) \models q$ for every $cens(\cdot) \in optGACens(\mathcal{H})$.

Our ultimate goal is to encode the above semantics in the OBDA framework, i.e. we want to know if it is possible to check if $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{CQ}} q$ or $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{GA}} q$ by verifying that q is entailed by an OBDA setting. To this aim, we define below the notion of query equivalence under censor between PPOBDA and OBDA specifications.

Definition 23. [Query equivalence] Let $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ be a PPOBDA specification, and let $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}' \rangle$ be an OBDA specification. We say that $\mathcal{H}$ and $\mathcal{J}$ are *query-equivalent under CQ censors (resp., GA censors)* if, for every database instance $\mathcal{D}$ for $\mathcal{S}$ and every BUCQ q, $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{CQ}} q$ (resp., $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{GA}} q$) iff $\langle \mathcal{J}, \mathcal{D} \rangle \models q$.

Based on the above definition, one can check whether $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{CQ}} q$ (resp., $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{GA}} q$) by constructing an OBDA specification $\mathcal{J}$ such that $\mathcal{H}$ and $\mathcal{J}$ are query-equivalent under CQ censors (resp., under GA censors) and verifying whether $\langle \mathcal{J}, \mathcal{D} \rangle \models q$. We remark that, besides the presence of the policy, the mapping is the only component in which $\mathcal{H}$ and $\mathcal{J}$ differ (see also Section 1). Intuitively, $\mathcal{M}'$ in $\mathcal{J}$ implements a censor (in either **CQ** or **GA**) for $\mathcal{H}$.

6.2 Inexpressibility results

In this section, we start investigating how to reduce query entailment in PPOBDA to query entailment in OBDA, based on the query equivalence definition given in the previous section.

Before proceeding further, we notice that, given a PPOBDA specification $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$, a natural question is whether the OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, i.e. obtained by simply eliminating the policy $\mathcal{P}$ from $\mathcal{H}$, is query-equivalent to $\mathcal{H}$ under censors either in **CQ** or in **GA**. In other terms, one might wonder whether the mapping $\mathcal{M}$ is already realizing a filter on the data such that denials in $\mathcal{P}$ are never violated by the underlying data retrieved through $\mathcal{M}$, whatever source database for $\mathcal{J}$ is considered[1]. If this would be the case, the entailment problems we are studying would become trivial. However, since the bodies of mapping assertions are FO queries, to answer the above question we should decide entailment in FO, which is an undecidable problem.

The following result says that, under CQ censors, constructing an OBDA specification that is query-equivalent to $\mathcal{H}$ is in general not possible, already for the case of an empty TBox, i.e. a TBox that does not contain axioms. As a consequence, entailment of BUCQs under CQ censors can not be solved through transformation in a query-equivalent OBDA specification, whatever logic is used for the TBox.

Theorem 8. *There exists a PPOBDA specification $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ with $\mathcal{T} = \emptyset$ for which there does not exist an OBDA specification $\mathcal{J}$ such that $\mathcal{H}$ and $\mathcal{J}$ are query-equivalent under CQ censors.*

Proof. Consider the PPOBDA specification $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ such that $\mathcal{T} = \emptyset$, $\mathcal{S}$ contains the binary relation R, $\mathcal{M} = \{R(x, y) \rightsquigarrow P(x, y)\}$, where $P \in \mathit{pred}(\mathcal{T})$, and $\mathcal{P}$ is as follows:

$$\mathcal{P} = \{\forall x\,(P(a, x) \rightarrow \bot), \quad \forall x\,(P(x, a) \rightarrow \bot)\},$$

where a is a constant from $\mathscr{C}$. Assume that $\mathcal{J}$ is an OBDA specification such that $\mathcal{H}$ and $\mathcal{J}$ are query-equivalent under CQ censors, and let $\mathcal{M}'$ be the mapping of $\mathcal{J}$. Consider now the case when the source database $\mathcal{D}$ consists of the fact $R(a, a)$. First, it is immediate to see that, given the policy $\mathcal{P}$, no BCQ mentioning the individual a can belong to $\mathit{cens}(\mathcal{D})$, for every censor $\mathit{cens}(\cdot) \in \mathit{optCQCens}(\mathcal{H})$. Then, since a is the only individual appearing in $\mathcal{D}$, it follows that no BCQ mentioning any individual can belong to $\mathit{cens}(\mathcal{D})$, for every censor $\mathit{cens}(\cdot) \in \mathit{optCQCens}(\mathcal{H})$. This implies that the mapping $\mathcal{M}'$ of $\mathcal{J}$ cannot retrieve any instance from $\mathcal{D}$, i.e. $\mathit{ret}(\mathcal{J}, \mathcal{D})$ is empty, and therefore no BUCQ is entailed by $\langle \mathcal{J}, \mathcal{D} \rangle$.

On the other hand, the OBDA setting $\langle \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, \mathcal{D} \rangle$ infers purely existential BUCQs. For instance, all the BCQs expressing existential cycles of any length over the role P, that is all the queries of the form:

$$\exists x_0, \ldots, x_n\,(P(x_0, x_1) \wedge P(x_1, x_2) \wedge \ldots \wedge P(x_n, x_0))$$

where $n \in \mathbb{N}$. All such queries can be positively answered by the PPOBDA setting $\langle \mathcal{H}, \mathcal{D} \rangle$ without revealing a secret: so, all such queries are returned by every censor

[1] Note that this is not the problem studied in [19] (see also the discussion in Chapter 10).

$cens(\cdot) \in optCQCens(\mathcal{H})$. Since they are not entailed by $\langle \mathcal{J}, \mathcal{D} \rangle$, this contradicts the hypothesis that $\mathcal{H}$ and $\mathcal{J}$ are query-equivalent under CQ censors, thus proving the theorem. ∎

Hereinafter, we focus on DL-Lite$_\mathcal{A}$ PPOBDA specifications, i.e. PPOBDA specifications whose TBox is expressed in the logic DL-Lite$_\mathcal{A}$. The following theorem states that the same issue of Theorem 8 arises also under GA censors.

Theorem 9. *There exists a DL-Lite$_\mathcal{A}$ PPOBDA specification $\mathcal{H}$ for which there does not exist an OBDA specification $\mathcal{J}$ such that $\mathcal{H}$ and $\mathcal{J}$ are query-equivalent under GA censors.*

Proof. We note that, for any CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, ABox $\mathcal{A}$ and BUCQ q, solving GA-Cens-Ent can always be reduced to checking whether $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{GA}} q$, with $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$. In particular, as already discussed, it is sufficient to take a mirrored database with respect to $\mathcal{A}$ and an identity mapping. As remarked in Proposition 4, however, the first problem is coNP-hard in data complexity, which allows us to claim the same lower bound also holds for the second one.

On the other hand, we point out that standard BUCQ entailment for OBDA specifications with a DL-Lite$_\mathcal{A}$ TBox is in AC^0 in data complexity [100]. This clearly implies the thesis. ∎

6.3 Embedding a policy into the mapping

Hereafter, we focus on the counterpart of the IGA censor for PPOBDA specifications, which is defined analogously. Moreover, given a PPOBDA specification $\mathcal{H}$, a source database $\mathcal{D}$, and a BUCQ q, we write $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{IGA}} q$ if $\mathcal{T} \cup cens_{\mathsf{IGA}}(\mathcal{D}) \models q$, where $cens_{\mathsf{IGA}}(\cdot)$ the IGA censor for $\mathcal{H}$. As in the case of CQE specifications, it is immediate to see that $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{IGA}} q$ implies $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{GA}} q$.

We also naturally extend Definition 23 to IGA censors. Given a PPOBDA specification $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ and an OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{M}', \mathcal{S} \rangle$, we say that $\mathcal{H}$ and $\mathcal{J}$ are *query-equivalent under IGA censor* if, for every source database $\mathcal{D}$ for $\mathcal{S}$ and every BUCQ q, $\langle \mathcal{H}, \mathcal{D} \rangle \models_{\mathsf{IGA}} q$ iff $\langle \mathcal{J}, \mathcal{D} \rangle \models q$.

In what follows, we prove that every DL-Lite$_\mathcal{A}$ PPOBDA specification $\mathcal{H}$ admits an OBDA specification $\mathcal{J}$ that is query-equivalent under IGA censor to $\mathcal{H}$, and provide an algorithm to build $\mathcal{J}$. The intuition behind our algorithm is as follows. For any source database $\mathcal{D}$, we want that $ret(\mathcal{J}, \mathcal{D})$ does not contain all those facts of $ret(\mathcal{H}, \mathcal{D})$ that together with the TBox $\mathcal{T}$ lead to the violation of the policy $\mathcal{P}$. At the same time, we want this elimination of facts to be done in a minimal way, according to our definition of IGA censor.

To this aim, one could think of exploiting the algorithm *MinPolicy* introduced in Section 3.3.1. Anyway, despite *MinPolicy* can be employed as a theoretical tool to establish the FO rewritability of the examined problem, one can see that it is not possible to use such a procedure in real-world situations, even in simple cases like the one illustrated in the following example.

Example 28. Consider a policy $\mathcal{P}$ consisting of the following two denials:

$$\forall x, y\,(C(x) \wedge R(x,y) \wedge S(y,a) \rightarrow \bot),$$
$$\forall x, y\,(R(x,x) \wedge S(x,y) \wedge S(y,a) \rightarrow \bot).$$

where $a \in \mathscr{C}$. One can verify that, with an empty TBox $\mathcal{T}$, the extended denials returned by *MinPolicy*$(\mathcal{P}, \mathcal{T})$ would be the following ones:

$$\begin{aligned}
&\forall x, y\,(C(x) \wedge R(x,y) \wedge S(y,a) \wedge x \neq y \wedge x \neq a \wedge y \neq a \rightarrow \bot),\\
&\quad \forall x\,(C(x) \wedge R(x,x) \wedge S(x,a) \wedge x \neq a \rightarrow \bot),\\
&\quad \forall x\,(C(x) \wedge R(x,a) \wedge S(a,a) \wedge x \neq a \rightarrow \bot),\\
&\quad \forall y\,(C(a) \wedge R(a,y) \wedge S(y,a) \wedge y \neq a \rightarrow \bot),\\
&\forall x, y\,(R(x,x) \wedge S(x,y) \wedge S(y,a) \wedge x \neq y \wedge x \neq a \wedge y \neq a \rightarrow \bot),\\
&\quad \forall x\,(R(x,x) \wedge S(x,x) \wedge S(x,a) \wedge x \neq a \rightarrow \bot),\\
&\quad \forall x\,(R(x,x) \wedge S(x,a) \wedge S(a,a) \wedge x \neq a \rightarrow \bot),\\
&\quad \forall y\,(R(a,a) \wedge S(a,y) \wedge S(y,a) \wedge y \neq a \rightarrow \bot),\\
&\qquad R(a,a) \wedge S(a,a) \rightarrow \bot.
\end{aligned}$$

Note that, already when $\mathcal{T} = \emptyset$, the number of denials contained in *MinPolicy*$(\mathcal{P}, \mathcal{T})$ is $O(|\mathcal{P}| \cdot 2^t)$, where t is the maximum number of distinct terms appearing in a denial. Clearly, in case the TBox is not empty, such a number would further escalate. ▽

We thus replace such an algorithm with a more efficient one called *PolicyRefine*, which is the one actually used in our practical implementation and for conducting our experiments. Before describing how *PolicyRefine* works, we need some preliminary definitions and properties.

First, we generalize the notion of extended denial assertion as a formula of the form $\forall \mathbf{x}\,(\phi(\mathbf{x}) \wedge \neg\pi(\mathbf{x}) \rightarrow \bot)$, where $\exists \mathbf{x}\,\phi(\mathbf{x})$ is a BCQ and $\pi(\mathbf{x})$ is a (possibly empty) disjunction of conjunctions of comparison atoms of the form $t_1 \,\mathrm{op}\, t_2$, where t_1 and t_2 are either variables occurring in $\mathbf{x}$ or constants in $\mathscr{C}$. An extended policy $\mathcal{P}'$ is a finite set of extended denials of this new form. We then say that an extended denial δ' is *non-redundant in* $\mathcal{P}'$ if every minimal (with respect to set inclusion) set of facts $\mathcal{F}$ such that $\{\delta\} \cup \mathcal{F}$ is inconsistent is also a minimal set of facts such that $\mathcal{P}' \cup \mathcal{F}$ is inconsistent. Given a policy $\mathcal{P}$, we aim to compute an extended policy $\mathcal{P}'$ where each denial $\delta \in \mathcal{P}$ is replaced with a non-redundant denial δ'. Throughout the subsequent discussion, we use the symbols $\mathcal{P}$ and $\mathcal{P}'$ accordingly.

Given a set Θ of comparison atoms and a comparison atom $\lambda = t_1 \,\mathrm{op}\, t_2$, we say that Θ *implies* λ (in symbols, $\Theta \models \lambda$) if, for every ground substitution σ of variables in $\mathit{vars}(\Theta) \cup \{t_1, t_2\}$, it holds $\sigma(\Theta) \models \sigma(\lambda)$. Given two sets of comparison atoms Θ and Θ', we write $\Theta \models \Theta'$ to denote that every atom in Θ' is implied by Θ. Moreover, given a denial $\delta = \forall \mathbf{x}\,(\phi(\mathbf{x}) \rightarrow \bot)$ and a set Θ of comparison atom, we define $\delta[\Theta]$ as the following denial:

$$\delta[\Theta] = \forall \mathbf{x}\left(\phi(\mathbf{x}) \wedge \left(\bigwedge_{t_1 \,\mathrm{op}\, t_2 \in \Theta(\delta)} t_1 \,\mathrm{op}\, t_2\right) \rightarrow \bot\right).$$

where $\Theta(\delta)$ is the set of comparison atoms $t_1 \,\mathrm{op}\, t_2$ such that $\Theta \models t_1 \,\mathrm{op}\, t_2$ and the terms t_1, t_2 are from $\mathscr{C} \cup \mathit{vars}(\delta)$. Intuitively, $\delta[\Theta]$ constrains δ to adhere to all the conditions expressed in Θ.

Definition 24. [Redundancy condition] Let δ and δ' be two distinct denials, and let Θ be a set of comparison atoms using only terms of δ and δ'. We say that Θ is a *redundancy condition for δ with respect to δ'* if $\delta' \models \delta[\Theta]$.

When it is not necessary to mention δ', if δ' belongs to a policy $\mathcal{P}$ we also call Θ a *redundancy condition for δ in $\mathcal{P}$*.

Example 29. Consider the policy $\mathcal{P}_1$ consisting of the following two denials:

$$\begin{aligned}\delta_1 &= \forall x, y, z, w\,(R(x,y) \wedge R(y,z) \wedge U(x,w) \wedge C(x) \rightarrow \bot),\\ \delta_2 &= \forall x', y', z'\ \ (R(x',x') \wedge U(x',y') \wedge y' > 2 \rightarrow \bot),\\ \delta_3 &= \forall x'', y''\ \ \ (U(x'',y'') \wedge y'' > 3 \rightarrow \bot).\end{aligned}$$

It is easy to see that the set $\Theta_1 = \{x = y, x = z, w > 2\}$ is a redundancy condition for δ_1 with respect to δ_2. Indeed, we have that $\delta_2 \models \delta_1[\Theta_1]$. Moreover, the set $\Theta_2 = \{y' > 3\}$ is a redundancy condition for δ_2 with respect to δ_3. On the other hand, there does not exist any redundancy condition for δ_3 in $\mathcal{P}_1$. ▽

Example 30. We now illustrate a case in which two denials are reciprocally redundant. Consider the policy $\mathcal{P}_2$ consisting of:

$$\begin{aligned}\delta_4 &= \forall x, y\ \ (R(x,x) \wedge S(x,y) \wedge S(y,a) \rightarrow \bot),\\ \delta_5 &= \forall x', y'\,(R(x',y') \wedge S(y',y') \wedge S(y',b) \rightarrow \bot).\end{aligned}$$

In this case, the sets $\Theta_4 = \{x = y, y = b\}$ and $\Theta_5 = \{x' = y', y' = a\}$ are, respectively, redundancy conditions for δ_4 and δ_5 in $\mathcal{P}_2$. ▽

Given a denial $\delta = \forall \mathbf{x}\,(\phi(\mathbf{x}) \rightarrow \bot)$ and a set RC of sets of comparison atoms, we denote by $notRed(\delta, RC)$ the extended denial:

$$notRed(\delta, RC) = \forall \mathbf{x}\left(\phi(\mathbf{x}) \wedge \bigwedge_{\Theta \in RC}\left(\neg \bigwedge_{\lambda \in \Theta(\delta)} \lambda\right) \rightarrow \bot\right)$$

By construction, the newly defined function satisfies the two properties below.

Proposition 14. *Let δ be a denial, let $\mathcal{F}$ be a set of facts, and let RC be a set of sets of comparison atoms. If $\mathcal{F}$ violates δ, then it also violates either $notRed(\delta, RC)$ or $\delta[\Theta]$, for some $\Theta \in RC$, but not both.*

Proposition 15. *Let $\mathcal{P}$ be a policy, let δ be a denial in $\mathcal{P}$, and let RC be a set of redundancy conditions for δ in $\mathcal{P}$. Then, the set $\mathcal{P} \setminus \{\delta\} \cup \{notRed(\delta, RC)\}$ is equivalent to $\mathcal{P}$.*

For technical reasons, we also define the auxiliary function $notRedAux(\delta, RC)$ that, given a denial $\delta = \forall \mathbf{x}\,(\phi(\mathbf{x}) \rightarrow \bot)$ and a set $RC = \{\Theta_1, \ldots, \Theta_n\}$ of sets of comparison atoms, returns the following set of denials:

$$notRedAux(\delta, RC) = \left\{\forall \mathbf{x}\left(\phi(\mathbf{x}) \wedge \bigwedge_{1 \leq i \leq n} \neg\lambda_i \rightarrow \bot\right) \;\middle|\; (\lambda_1, \ldots, \lambda_n) \in \Theta_1 \times \cdots \times \Theta_n\right\}$$

where each $\neg\lambda_i$ is replaced with the corresponding negated form of the comparison atom. For instance, if $\delta = \forall x, y\,(R(x,y) \rightarrow \bot)$ and $RC = \{\{x = y\}, \{x > 1, y \neq 3\}\}$, then $\mathit{notRedAux}(\delta, RC)$ returns the following denials:

$$\forall x, y\,(R(x,y) \wedge x \neq y \wedge x \leq 1 \rightarrow \bot)$$
$$\forall x, y\,(R(x,y) \wedge x \neq y \wedge y = 3 \rightarrow \bot)$$

It is important to notice that the set returned by $\mathit{notRedAux}(\delta, RC)$ is logically equivalent to the singleton $\{\mathit{notRed}(\delta, RC)\}$. Indeed, it can be computed by distributing the (implicit) disjunction in $\mathit{notRed}(\delta, RC)$ over the outer conjunction, and then creating a different denial for each member of the resulting disjunction.

We are now able to define the procedure *PolicyRefine* (Algorithm 2). Given a policy $\mathcal{P}$, $\mathit{PolicyRefine}(\mathcal{P})$ scans the denials δ occurring in $\mathcal{P}$ and compares them with all the denials δ' that either follow δ in the scanning order or are "non-redundant" versions of some denial preceding δ. For each pair of distinct denials δ and δ' considered, all the possible containments between the CQs in the premises of δ and δ' are checked by exhaustively examining all the possible partitions of their sets of predicate atoms into the subsets Q_i and Q'_i respectively (we remark that every such subset of atoms Q_i and Q'_i must be non-empty), and then checking whether such subsets of atoms are pairwise unifiable. In this way, the algorithm computes a *complete* set of redundancy conditions RC for δ with respect to δ', meaning that, for every possible redundancy condition Θ' for δ with respect to δ', there exists a set $\Theta \in RC$ such that $\Theta' \models \Theta$. Observe that, towards this aim, we search for an MGU between the subsets of atoms of δ and δ' (a simple unifier would not guarantee such a completeness property). In the algorithm, we assume w.l.o.g. that the unifier returned by the $MGU(\cdot)$ function maps each term either to a constant in $\mathscr{C}$ or to a variable of δ. Moreover, given a substitution σ and a denial δ, the function $\mathit{eqs}(\sigma, \delta)$ returns a minimal (w.r.t. set inclusion) set Θ of equalities such that $\sigma(\delta)$ is equivalent to $\delta[\Theta]$. For example, if $\delta = \forall x, y\,(R(x,y) \wedge S(y,z) \rightarrow \bot)$ and $\sigma = \{w \mapsto x, y \mapsto x, z \mapsto 1\}$, then $\mathit{eqs}(\sigma, \delta) = \{x = y, z = 1\}$.

Example 31. Consider the policy $\mathcal{P}_1$ of Example 29, and assume that the denials are processed by *PolicyRefine* in the given order (i.e. $\langle\delta_1, \delta_2, \delta_3\rangle$). The output of $\mathit{PolicyRefine}(\mathcal{P}_1)$ is:

$$\begin{aligned}
\delta'_1 &= \forall x, y, z, w\,(\,R(x,y) \wedge R(y,z) \wedge U(x,w) \wedge C(x) \wedge \\
&\qquad\qquad \neg(x = y \wedge x = z, w > 2) \wedge \neg(w > 3) \rightarrow \bot),\\
\delta'_2 &= \forall x', y', z' \quad (\,R(x',x') \wedge U(x',y') \wedge y' > 2 \wedge \neg(y' > 3) \rightarrow \bot),\\
\delta'_3 &= \forall x'', y'' \quad\;\; (\,U(x'',y'') \wedge y'' > 3 \rightarrow \bot).
\end{aligned}$$

Consider now the policy $\mathcal{P}_2$ of Example 30, and suppose that δ_4 is processed before δ_5. The output of $\mathit{PolicyRefine}(\mathcal{P}_2)$ would be:

$$\begin{aligned}
\delta'_4 &= \forall x, y \quad (\,R(x,x) \wedge S(x,y) \wedge S(y,a) \wedge \neg(x = y \wedge y = b) \rightarrow \bot),\\
\delta'_5 &= \forall x', y' \,(\,R(x',y') \wedge S(y',y') \wedge S(y',b) \wedge \\
&\qquad \neg(x' = y' \wedge y' = a \wedge y' \neq y') \wedge \neg(x' = y' \wedge y' = a \wedge b \neq b) \rightarrow \bot)
\end{aligned}$$

(note that δ'_5 is logically equivalent to δ_5, since the two negated subformulas are self-contradictory). In fact, δ_5 is compared with the two denials in $\mathit{notRedAux}(\delta_4, RC)$,

Algorithm 2 *PolicyRefine*

input: a policy $\mathcal{P}$;
output: an extended policy $\mathcal{P}'$;
1: $\mathcal{P}' \leftarrow \emptyset$;
2: $\mathcal{P}^{\mathsf{aux}} \leftarrow \emptyset$;
3: **for each** denial $\delta \in \mathcal{P}$ **do**
4: $RC \leftarrow \emptyset$;
5: $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\delta\}$
6: **for each** denial $\delta' \in \mathcal{P} \cup \mathcal{P}^{\mathsf{aux}}$ **do**
7: **for each** non-empty set $\Phi \subseteq \mathit{predAt}(\delta)$ **do**
8: **for each** partition $Q_1, \ldots, Q_k$ of Φ **do**
9: **for each** partition $Q'_1, \ldots, Q'_k$ of $\mathit{predAt}(\delta')$ **do**
10: **if** $Q_i \cup Q'_i$ is a set of unifiable atoms for every i s.t. $1 \leq i \leq k$ **then**
11: $\sigma \leftarrow \bigcup_{1 \leq i \leq k} \mathit{MGU}(Q_i \cup Q'_i)$;
12: $RC \leftarrow RC \cup \{\mathit{eqs}(\sigma, \delta) \cup \mathit{compAt}(\sigma(\delta'))\}$;
13: $\mathcal{P}^{\mathsf{aux}} \leftarrow \mathcal{P}^{\mathsf{aux}} \cup \mathit{notRedAux}(\delta, RC)$;
14: $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{\mathit{notRed}(\delta, RC)\}$;
15: **return** $\mathcal{P}'$;

namely:

$$\delta_{\mathsf{aux}} = \forall x, y\,(R(x,x) \wedge S(x,y) \wedge S(y,a) \wedge x \neq y \rightarrow \bot),$$
$$\delta'_{\mathsf{aux}} = \forall x, y\,(R(x,x) \wedge S(x,y) \wedge S(y,a) \wedge y \neq b \rightarrow \bot).$$

One can verify that, if δ_5 is processed before δ_4, then the opposite would happen (i.e. δ'_4 would be equivalent to δ_4). ▽

Lemma 6. *Let $\mathcal{P}$ be a policy, and let $\mathcal{P}' = \mathit{PolicyRefine}(\mathcal{P})$. Then, $\mathcal{P}'$ is equivalent to $\mathcal{P}$.*

Proof. Let us use the j subscript for denoting the (final) value of any variable used in the algorithm at the j-th iteration of the outermost loop. Let also $\widetilde{\mathcal{P}}_j$ denote the value of $\mathcal{P}_j \cup \mathcal{P}^{\mathsf{aux}}_j$ computed at Step 6, and let $n = |\mathcal{P}|$.

It is easy to see that RC_i is a set of redundancy conditions for δ_i in $\widetilde{\mathcal{P}}_i$. Then, by Proposition 15 we have that $\widetilde{\mathcal{P}}_i \cup \{\delta_i\}$ is equivalent to $\widetilde{\mathcal{P}}_i \cup \{\mathit{notRed}(\delta_i, RC_i)\}$ (or, equivalently, to $\widetilde{\mathcal{P}}_i \cup \mathit{notRedAux}(\delta_i, RC_i)$), for every i such that $1 \leq i \leq n$. Note also that:

- $\widetilde{\mathcal{P}}_1 \cup \{\delta_1\} = \mathcal{P}$;
- $\widetilde{\mathcal{P}}_i \cup \mathit{notRedAux}(\delta_i, RC_i)$ is equal to $\widetilde{\mathcal{P}}_{i+1} \cup \{\delta_{i+1}\}$ (i.e. $\widetilde{\mathcal{P}}_i \cup \{\delta_i\}$ is equivalent to $\widetilde{\mathcal{P}}_{i+1} \cup \{\delta_{i+1}\}$) for each i such that $1 \leq i \leq n-1$;
- $\widetilde{\mathcal{P}}_n = \mathcal{P}^{\mathsf{aux}}_{n-1}$ (because $\mathcal{P}_n = \emptyset$).

Thus, the input $\mathcal{P}$ is equivalent to $\mathcal{P}^{\mathsf{aux}}_{n-1} \cup \mathit{notRedAux}(\delta_n, RC_n)$, i.e. to $\mathcal{P}^{\mathsf{aux}}_n$, which in turn is equivalent to the output $\mathcal{P}'$. ∎

Lemma 7. *Let $\mathcal{P}$ be a policy, and let $\mathcal{P}' = \mathit{PolicyRefine}(\mathcal{P})$. Then, every $\delta' \in \mathcal{P}'$ is non-redundant in $\mathcal{P}'$.*

Proof. Let $\mathcal{F}$ be a minimal (w.r.t. set inclusion) set of facts such that $\mathcal{F}$ violates δ'_1. We show that every proper subset $\mathcal{F}' \subset \mathcal{F}$ satisfies every $\delta'_2 \in \mathcal{P}'$. Actually, we show a stronger property, i.e. that $\mathcal{F}$ itself does not violate any denial in $\delta'_2 \in \mathcal{P}' \setminus \{\delta'_1\}$.

By way of contradiction, let there exist an extended denial $\delta'_2 \in \mathcal{P}' \setminus \{\delta'_1\}$ such that $\mathcal{F}$ violates δ'_2. Let δ_1 and δ_2 be the corresponding denials from $\mathcal{P}$ for which $\delta'_1 = \mathit{notRed}(\delta_1, RC_1)$ and $\delta'_2 = \mathit{notRed}(\delta_2, RC_2)$. Furthermore, let us fix any pair of instantiations σ_1^* and σ_2^* of the premises of δ'_1 and δ'_2 in $\mathcal{F}$, respectively.

Note that $\sigma_1^* \cup \sigma_2^*$ unifies all the predicate atoms occurring in δ'_2 with (a subset of) the ones occurring in δ'_1, if suitably coupled. This disparity comes from the fact that $\mathcal{F}$ is a minimal set of facts violating δ'_1 but a non-necessarily minimal set of facts violating δ'_2. Note also that, by construction, all the comparison atoms occurring in the ground denial $\sigma_2^*(\delta_2)$ are tautologies, thus we can assume w.l.o.g. that it consists of a conjunction of predicate atoms.

Now, δ_1 could have been selected before or after δ_2 in Step 3 of Algorithm 2.

1. In the first case, δ_2 is chosen from $\mathcal{P} \cup \mathcal{P}^{\mathsf{aux}}$ at Step 6. Note that Step 11 computes a union of MGUs for every possible way of unifying the predicate atoms of δ_2 with (a subset of) the ones of δ_1. In particular, by construction of σ_1^* and σ_2^*, at least one of such unions of MGUs must be extended by $\sigma_1^* \cup \sigma_2^*$. Let us call $\Theta \in RC_1$ the redundancy condition computed by *PolicyRefine* in the case of such an MGU. It is easy to see that $\mathcal{F}$ violates $\delta_1[\Theta]$, which by Proposition 14 implies that $\delta'_1 = \mathit{notRed}(\delta_1, RC_1)$ is not violated by $\mathcal{F}$ (a contradiction).

2. In the second case, we have two possibilities:

   - Some MGU σ found in the inner loop can be extended in such a way to behave as σ_2^* when applied to δ_2. In this case, as before, the extended denial $\delta'_2 = \mathit{notRed}(\delta_2, RC_2)$ can not be violated by $\mathcal{F}$, resulting again in a contradiction.
   - No such an MGU σ is found in the inner loop. In this case, some denial $\delta''_2 \in \mathit{notRedAux}(\delta_2, RC_2)$ is still violated by $\mathcal{F}$. At the iteration in which δ_1 is selected in Step 3, δ''_2 will be selected at Step 6 from $\mathcal{P} \cup \mathcal{P}^{\mathsf{aux}}$, which leads us back to Point 1. ∎

Example 32. Consider CQE specification $\langle \mathcal{T}, \mathcal{P} \rangle$ provided in Example 28. One can verify that *PolicyRefine*(*PerfectRef*$(\mathcal{P}, \mathcal{T})$) returns the following set of denials:

$$\forall x, y\, (C(x) \wedge R(x, y) \wedge S(y, a) \wedge \neg(x = y \wedge y = a) \rightarrow \bot),$$
$$\forall x, y\, (R(x, x) \wedge S(x, y) \wedge S(y, a) \rightarrow \bot).$$

We observe that such a set is significantly simpler than the one resulting in Example 28. ▽

Hereinafter, given a policy $\mathcal{P}$ and a DL-Lite$_\mathcal{A}$ TBox $\mathcal{T}$, we refer to the expansion $\mathcal{P}$ with respect to $\mathcal{T}$ by using the function *expandPol*$(\mathcal{P}, \mathcal{T})$ which returns the following set of denials:

$$\mathit{expandPol}(\mathcal{P}, \mathcal{T}) = \Big\{ q' \rightarrow \bot \mid q' \in \mathit{PerfectRef}\big(\bigvee_{q \rightarrow \bot \in \mathcal{P}} q, \mathcal{T}\big) \Big\}.$$

The following proposition follows from the definition of satisfaction of a denial and from Proposition 1.

Proposition 16. *Let $\mathcal{T} \cup \mathcal{A}$ be a consistent DL-Lite$_\mathcal{R}$ ontology, and let $\mathcal{P}$ be a policy. Then, $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}$ is a consistent FO theory iff $expandPol(\mathcal{P}, \mathcal{T}) \cup \mathcal{A}$ is a consistent FO theory.*

The following theorem shows that the algorithms *PolicyRefine* and *PerfectRef* can be combined for generating a non-redundant representation of a given policy with respect to a DL-Lite$_\mathcal{A}$ TBox.

Theorem 10. *Let $\langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{A}$ CQE specification, let $\hat{\mathcal{P}} = expandPol(\mathcal{P}, \mathcal{T})$, and let $\mathcal{P}'$ be the extended policy returned by $PolicyRefine(\hat{\mathcal{P}})$. Then, $\mathcal{P}'$ is a non-redundant representation of $\mathcal{P}$ with respect to $\mathcal{T}$.*

Proof. Condition (i) of Definition 11 immediately follows from Lemma 6 and Proposition 16. We now show that each $\delta' = notRed(\delta, RC_\delta)$ returned by *PolicyRefine* also satisfies condition (ii) of the same definition. Let $\mathcal{A}$ be any ABox for $\mathcal{T}$. If $\mathcal{F} \subseteq cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ is a minimal set of facts such that $\{\delta'\} \cup \mathcal{F}$ is inconsistent, then by Lemma 7, $\mathcal{F}$ is a minimal set of facts such that $\mathcal{P}' \cup \mathcal{F}$ is inconsistent. Since, by Lemma 6, $\mathcal{P}'$ is equivalent to $\hat{\mathcal{P}}$, then $\mathcal{F}$ is also a minimal set of facts such that $\hat{\mathcal{P}} \cup \mathcal{F}$ is inconsistent or, equivalently, such that $\mathcal{T} \cup \mathcal{P} \cup \mathcal{F}$ is inconsistent. By Definition 10, we thus have that $\mathcal{F} \in secrets(\langle \mathcal{T}, \mathcal{P} \rangle, \mathcal{A})$. ∎

We recall that we avoid retrieving the "dangerous facts" by embedding suitable conditions in the mapping assertions of $\mathcal{M}'$. This is done by the algorithm *PolicyEmbed*, which is outlined below.

Algorithm 3 *PolicyEmbed*

input: a DL-Lite$_\mathcal{A}$ TBox $\mathcal{T}$, a mapping $\mathcal{M}$, a policy $\mathcal{P}$;
output: a mapping $\mathcal{M}'$;
1: $\mathcal{P}' \leftarrow PolicyRefine(expandPol(\mathcal{P}, \mathcal{T}))$;
2: $\mathcal{M}' \leftarrow \emptyset$;
3: **for each** predicate $P \in pred(\mathcal{T})$ **do**
4: **let** $\mathbf{x}$ be a k-tuple of variables, where k is the arity of P
5: $\phi \leftarrow AtomRewr(addPolicyConditions(P(\mathbf{x}), \mathcal{P}'), \mathcal{T})$
6: $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{unfold(\phi, \mathcal{M}) \rightsquigarrow P(x)\}$
7: **return** $\mathcal{M}'$;

It follows a detailed description of Algorithm 3.

- Step 1 produces the non-redundant policy $\mathcal{P}'$ by means of $expandPol(\cdot, \cdot)$ and $PolicyRefine(\cdot)$.
- Step 2 initializes an empty mapping, $\mathcal{M}'$, which will be subsequently populated with one mapping assertion for each ontology predicate.
- Step 5 generates the rewriting for a single atom $P(\mathbf{x})$ by means of the functions $addPolicyConditions(\cdot, \cdot)$ (that is a simplified version of the $Discl(\cdot, \cdot)$ function

exploiting the fact that $\mathcal{P}'$ is already non-redundant with respect to $\mathcal{T}$) and *AtomRewr*$(\cdot,\cdot)$. Formally, *addPolicyConditions*$(\alpha, \mathcal{P}')$ is defined as follows:

$$addPolicyConditions(\alpha, \mathcal{P}') = \alpha \wedge \bigwedge_{\substack{q_\delta \in \mathcal{P}', \\ \sigma \in findSub(\alpha, q_\delta)}} \forall \mathbf{w}(\neg\sigma(q_\delta))$$

- At Step 6, the mapping is incremented with the mapping assertion for P. The function *unfold*$(\cdot,\cdot)$ realizes a typical unfolding for GAV mapping [120], i.e. it substitutes each atom α with the union of the body of all mapping assertions having α in their heads. Unfolding also the negative part of ϕ guarantees that no fact causing a violation of a denial involving C is retrieved.

The next theorem states that *PolicyEmbed* can be used to obtain the PPOBDA-to-OBDA transformation we were looking for.

Theorem 11. *Let $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{A}$ PPOBDA specification, and let $\mathcal{J}$ be the OBDA specification $\langle \mathcal{T}, \mathcal{S}, \mathcal{M}' \rangle$, where $\mathcal{M}'$ is the mapping returned by PolicyEmbed$(\mathcal{T}, \mathcal{M}, \mathcal{P})$. Then, $\mathcal{J}$ is query-equivalent to $\mathcal{H}$ under IGA censor.*

Proof. Let $\mathcal{D}$ be a source database for $\mathcal{S}$. We prove the theorem by showing that *ret*$(\mathcal{J}, \mathcal{D})$ is equal to *cens*$_{\mathsf{IGA}}(\mathcal{D})$.

Let α be any ground atom. We recall that, by Theorem 10, the extended policy $\mathcal{P}' =$ *PolicyRefine*(*expandPol*$(\mathcal{P}, \mathcal{T})$) is a non-redundant representation of $\mathcal{P}$ with respect to $\mathcal{T}$. Then, by employing analogous considerations to those in the proof of Theorem 2, one can see that evaluating in *ret*$(\mathcal{H}, \mathcal{D})$ the FO query $\phi_\alpha =$ *AtomRewr*(*addPolicyConditions*$(\alpha, \mathcal{P}')$) decides if α evaluates to true (i.e. it is contained) in *cens*$_{\mathsf{IGA}}(\mathcal{D})$.

On the other hand, the mapping assertions returned by *PolicyEmbed* are built by unfolding the query ϕ (associated with a specific predicate) with respect to the original mapping $\mathcal{M}$ (Step 6). Therefore, we have that *unfold*$(\phi_\alpha, \mathcal{M}) =$ *unfold*$(\alpha, \mathcal{M}')$, from which it follows that, for every database $\mathcal{D}$, $\langle \mathcal{H}, \mathcal{D} \rangle \models \phi_\alpha$ iff $\langle \mathcal{J}, \mathcal{D} \rangle \models \alpha$. In other words, we have that ϕ_α evaluates to true in *ret*$(\mathcal{H}, \mathcal{D})$ iff α evaluates to true (i.e. it is contained) in *ret*$(\mathcal{J}, \mathcal{D})$.

Since both *cens*$_{\mathsf{IGA}}(\cdot)$ and *ret*$(\cdot,\cdot)$ return sets of ground atoms, this is possible only if *cens*$_{\mathsf{IGA}}(\mathcal{D}) =$ *ret*$(\mathcal{J}, \mathcal{D})$. ∎

Example 33. Let $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ and $\mathcal{D}$ be, respectively, the PPOBDA specification and source database of Example 27. If *cens*$_{\mathsf{IGA}}(\cdot)$ is the IGA censor for $\mathcal{H}$, then we have that:

$$\begin{aligned} cens_{\mathsf{IGA}}(\mathcal{D}) = \{\ & \mathsf{Person(ann)}, \mathsf{Person(bob)}, \mathsf{age(bob, 59)}, \mathsf{Antidysuric(m}_b\mathsf{)}, \\ & \mathsf{Medicine(m}_a\mathsf{)}, \mathsf{Medicine(m}_b\mathsf{)}, \mathsf{Product(m}_a\mathsf{)}, \mathsf{Product(m}_b\mathsf{)}, \\ & \mathsf{buy(bob, m}_b\mathsf{)}\ \} \end{aligned}$$

Let now $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}' \rangle$ be the OBDA specification such that $\mathcal{M}'$ be the set of

mapping assertions returned by *PolicyEmbed*$(\mathcal{T}, \mathcal{M}, \mathcal{P})$, i.e.:

$\exists a, w\,(\texttt{person}(p, a, w)) \vee \exists t, pr(\texttt{transaction}(t, p, pr)) \rightsquigarrow \mathsf{Person}(p)$,
$\exists w\,(\texttt{person}(p, a, w) \wedge \neg\exists t, m, e, b, c\,(a > 65 \wedge \texttt{transaction}(t, p, m) \wedge$
$\texttt{medicine}(m, e, b, c) \wedge \texttt{api}(b, \texttt{`phenazopyridine'}))) \rightsquigarrow \mathsf{age}(p, a)$,
$\exists t\,(\texttt{transaction}(t, p, m) \wedge \neg\exists a, w, e, b, c(\texttt{person}(p, a, w) \wedge a > 65 \wedge$
$\texttt{medicine}(m, e, b, c) \wedge \texttt{api}(b, \texttt{`phenazopyridine'}))) \rightsquigarrow \mathsf{buy}(p, m)$,
$\exists e, b, c\,(\texttt{medicine}(m, e, b, c) \wedge \texttt{api}(b, \texttt{`phenazopyridine'}) \wedge \neg\exists p, a, w, t(a > 65 \wedge$
$\texttt{person}(p, a, w) \wedge \texttt{transaction}(t, p, m))) \rightsquigarrow \mathsf{Antidysuric}(m)$,
$\exists e, b, c\,(\texttt{medicine}(m, e, b, c)) \vee \exists e, b, c\,(\texttt{medicine}(m, e, b, c) \wedge$
$\texttt{api}(b, \texttt{`phenazopyridine'})) \rightsquigarrow \mathsf{Medicine}(m)$,
$\exists t, p\,(\texttt{transaction}(t, p, pr)) \vee \exists e, b, c\,(\texttt{medicine}(pr, e, b, c)) \vee$
$\exists e, b, c\,(\texttt{medicine}(pr, e, b, c) \wedge \texttt{api}(b, \texttt{`phenazopyridine'})) \rightsquigarrow \mathsf{Product}(pr)$,

One can verify that $ret(\mathcal{J}, \mathcal{D}) = cens_{\mathsf{IGA}}(\mathcal{D})$. ▽

In the last part of this section, we adapt the above approach to prioritized CQE. Specifically, we show that the same result above can be extended to reasoning under the k-DD censor.

We call *prioritized PPOBDA specification* the quintuple $\mathcal{H}_\succ = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P}, \succ \rangle$, where $\langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ is a PPOBDA specification and $\succ$ is a priority relation over the signature of $\mathcal{T}$. The definition of k-DD censor for a prioritized PPOBDA specification $\mathcal{H}_\succ$ is analogous to Definition 18. Moreover, given a prioritized PPOBDA specification $\mathcal{H}$, a source database $\mathcal{D}$, a positive integer k, and BUCQ q, we write $\langle \mathcal{H}_\succ, \mathcal{D} \rangle \models^k_{\mathsf{DD}} q$ if $\mathcal{T} \cup cens_{\mathsf{DD_k}}(\mathcal{D}) \models q$, where $cens_{\mathsf{DD_k}}(\cdot)$ the k-DD censor for $\mathcal{H}_\succ$.

As in the case of IGA censor, we also adapt the notion of query equivalence first given in Definition 23. Given a prioritized PPOBDA specification $\mathcal{H}_\succ = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P}, \succ \rangle$ and an OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{M}', \mathcal{S} \rangle$, we say that $\mathcal{H}_\succ$ and $\mathcal{J}$ are *query-equivalent under k-DD censor* if, for every source database $\mathcal{D}$ for $\mathcal{S}$ and every BUCQ q, $\langle \mathcal{H}_\succ, \mathcal{D} \rangle \models^k_{\mathsf{DD}} q$ iff $\langle \mathcal{J}, \mathcal{D} \rangle \models q$.

Let us now consider a modified version of the algorithm *PolicyEmbed*, which takes as input also the relation $\succ$ and a positive integer k, and behaves like the original except for Step 5. This step, indeed, relies on a simplified version of the rewriting function $DD_k(\cdot, \cdot)$, in the same way the original algorithm was adopting a simplified version of $Discl(\cdot, \cdot)$ (i.e. by directly using the denials in $\mathcal{P}'$ instead of invoking the procedure *MinPolicy*). In other words, in such a step we use a function *addPolicyConditions*$(\alpha, \mathcal{P}', k)$ that is defined as $DD_k(\alpha, \mathcal{P}')$ except that $\mathcal{Q}_\mathcal{P}$ is simply the set of premises of the denials in $\mathcal{P}'$.

We are now ready to prove the theorem below.

Theorem 12. *Let $\mathcal{H} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{A}$ PPOBDA specification, let $\succ$ be a priority relation over pred$(\mathcal{T})$, and let $\mathcal{J}$ be the OBDA specification $\langle \mathcal{T}, \mathcal{S}, \mathcal{M}' \rangle$, where $\mathcal{M}'$ is the mapping returned by PolicyEmbed$(\mathcal{T}, \mathcal{M}, \mathcal{P}, \succ, k)$. Then, $\mathcal{J}$ is query-equivalent to $\mathcal{H}$ under k-DD censor with respect to $\succ$.*

Proof. Note that all the considerations done in the proof of Theorem 11 still hold. In particular we observe that, for every ground atom α, the formula computed by (the modified version of) *PolicyEmbed* in Step 5 is semantically equivalent to

$DDRew(\alpha, \mathcal{E}_{\succ}, k)$, i.e. it can be used to decide if α evaluates to true (i.e. is contained) in $cens_{\mathsf{DD_k}}(\mathcal{D})$. ∎

6.4 Experiments

In this section, we report the results of the experimentation we carried out using the NPD benchmark for OBDA [79]. The benchmark is based on real data coming from the oil industry: the Norwegian Petroleum Directorate (NPD) FactPages,[2] which were made available in 2013 [109]. It provides an OWL 2 ontology,[3] the NPD database, the mapping between the ontology and the database, an RDF [75] file specifying the instances of the ontology predicates, i.e. the retrieved ABox of the OBDA setting, and a set of 30 queries formulated in SPARQL [101].

For our experimentation, we produced (similarly to [53]) an approximation in OWL 2 QL of the OWL 2 benchmark ontology. Moreover, we made use of the benchmark RDF file containing the retrieved ABox to populate a relational database constituted by unary and binary tables (a unary table for each concept of the ontology and a binary table for each role and each attribute). Finally, we specified a mapping between the ontology and this database. In this case, the mapping in simply a set of one-to-one mapping assertions, i.e. every ontology predicate is mapped to the database table containing its instances. This kind of OBDA specification, with the simplest possible form of mapping assertions, allowed us to verify the feasibility of our technique for data protection, abstracting us from the impact of more complex queries in the original mapping.

In the resulting OBDA setting, the TBox comprises 1377 axioms over 321 atomic concepts, 135 roles, and 233 attributes. There are in total 2 million instances circa, which are stored in a MySQL database of 689 tables.

As for the queries, we considered nine (non-Boolean) $\text{CQ}^{\neq}$s from the ones provided with the NPD benchmark. In order to show the variability of query results depending on the specific considered setting (as described below), we kept all such queries open, i.e. we did not add any existential quantifier for their free variables. We remark that the entailment problems that we considered can be naturally extended to open queries. Strictly speaking, however, some of these queries from the benchmark are not $\text{CQ}^{\neq}$s, since they use aggregation operators, so we have extracted from them their conjunctive subqueries. Furthermore, we made use of the `DISTINCT` SPARQL keyword where missing in such queries, in order to properly compare the returned answer sets. We list below the resulting queries.

q_1 : $\exists li\,(\mathsf{ProductionLicence}(li) \wedge \mathsf{name}(li, ln) \wedge \mathsf{dateLicenceGranted}(li, d) \wedge \mathsf{isActive}(li, a) \wedge \mathsf{licensingActivityName}(li, an))$

q_2 : $\exists li, w\,(\mathsf{ProductionLicence}(li) \wedge \mathsf{explorationWellboreForLicence}(w, li) \wedge \mathsf{dateWellboreEntry}(w, e) \wedge \mathsf{name}(li, n))$

q_3 : $\exists le, li, c\,(\mathsf{licenseeForLicence}(le, li) \wedge \mathsf{ProductionLicence}(li) \wedge \mathsf{name}(li, ln) \wedge \mathsf{licenceLicensee}(le, c) \wedge \mathsf{name}(c, n) \wedge \mathsf{dateLicenseeValidFrom}(le, d))$

[2] `https://factpages.sodir.no/`

[3] `http://sws.ifi.uio.no/vocab/npd-v2`

q_4 : $\exists li, w\,(\mathsf{ProductionLicence}(li) \wedge \mathsf{name}(li, n) \wedge \mathsf{belongsToWell}(w, we) \wedge \mathsf{explorationWellboreForLicence}(w, li) \wedge \mathsf{name}(we, wn))$

q_5 : $\exists s, lu, c\,(\mathsf{wellboreStratumTopDepth}(s, st) \wedge \mathsf{inLithostratigraphicUnit}(s, lu) \wedge \mathsf{stratumForWellbore}(s, w) \wedge \mathsf{name}(w, n) \wedge \mathsf{wellboreStratumBottomDepth}(s, sb)\ \mathsf{name}(lu, un) \wedge \mathsf{WellboreCore}(c) \wedge \mathsf{coreForWellbore}(c, w) \wedge \mathsf{coreIntervalTop}(c, ct) \wedge \mathsf{coreIntervalBottom}(c, cb))$

q_6 : $\exists wc, w, c\,(\mathsf{WellboreCore}(wc) \wedge \mathsf{coreForWellbore}(wc, w) \wedge \mathsf{name}(w, wn) \wedge \mathsf{Wellbore}(w) \wedge \mathsf{wellboreCompletionYear}(w, y) \wedge \mathsf{drillingOperatorCompany}(w, c) \wedge \mathsf{name}(c, cn))$

q_7 : $\exists w, c\,(\mathsf{Wellbore}(w) \wedge \mathsf{name}(w, n) \wedge \mathsf{wellboreCompletionYear}(w, y) \wedge \mathsf{drillingOperatorCompany}(w, c) \wedge \mathsf{name}(c, cn))$

q_8 : $\exists p, m, f, op\,(\mathsf{productionYear}(p, 2010) \wedge \mathsf{productionMonth}(p, m) \wedge \mathsf{Field}(f) \wedge \mathsf{producedGas}(p, g) \wedge \mathsf{producedOil}(p, o) \wedge \mathsf{productionForField}(p, f) \wedge \mathsf{name}(f, fn) \wedge \mathsf{currentFieldOperator}(f, op) \wedge \mathsf{shortName}(op, \texttt{"STATOIL PETROLEUM AS"}))$

q_9 : $\exists y, f, c\,(\mathsf{coreForWellbore}(c, w) \wedge \mathsf{wellboreAgeTD}(w, a) \wedge \mathsf{wellboreEntryYear}(w, y) \wedge \mathsf{explorationWellboreForField}(w, f) \wedge \mathsf{Field}(f) \wedge \mathsf{name}(f, n))$

We ran each query in five different settings. In the first one we set an empty policy (and, consequently, an empty priority relation), which corresponds to the case of standard query answering over the uncensored ontology. For the other settings, we specified a policy $\mathcal{P}$ constituted by the denials enumerated below:

d_1 : $\forall w, d, i\,(\mathsf{dateWellboreEntry}(w, d) \wedge \mathsf{wellboreMaxInclation}(w, i) \wedge \mathsf{wellboreType}(w, \texttt{"initial"}) \wedge i \neq 6 \rightarrow \bot)$

d_2 : $\forall c, w, d, y\,(\mathsf{coreForWellbore}(c, w) \wedge \mathsf{wellboreCompletionYear}(w, y) \wedge \mathsf{documentForWellbore}(d, w) \wedge y \neq 1985 \rightarrow \bot)$

d_3 : $\forall w, c, t, s\,(\mathsf{wellOperator}(w, c) \wedge \mathsf{taskForCompany}(t, c) \wedge \mathsf{wellboreCompletionYear}(w, 1985) \wedge \mathsf{oilSampleTestForWellbore}(s, w) \rightarrow \bot)$

d_4 : $\forall w, l, d\,(\mathsf{explorationWellboreForLicence}(w, l) \wedge \mathsf{documentForWellbore}(d, w) \rightarrow \bot)$

d_5 : $\forall f, p, l\,(\mathsf{Field}(f) \wedge \mathsf{currentFieldOwner}(f, p) \wedge \mathsf{ProductionLicence}(p) \wedge \mathsf{licenseeForLicence}(l, p) \rightarrow \bot)$

d_6 : $\forall p, f\,(\mathsf{productionMonth}(p, 1) \wedge \mathsf{productionForField}(p, f) \rightarrow \bot)$

By coupling $\mathcal{P}$ with the OWL 2 QL version of the NPD TBox we obtained a safe DL-Lite$_\mathcal{A}$ CQE specification. In the second setting, the prioritized CQE specification contains the policy $\mathcal{P}$ illustrated above, and an empty priority relation, which corresponds, according to Proposition 10, to the IGA case. All the other settings are intended to verify the effectiveness of providing a priority relation $\succ$ and filtering data with a k-DD censor. In each setting we used a different odd k with $1 \leq k \leq 5$,[4] and considered the following priority relations, which, together with the denials in

[4]It's worth noting that, by construction of the k-DD censor, for every $k > 1$ we would obtain different results from the $k - 1$ case only for odd values of k.

query	metric	setting				
		$\emptyset, \emptyset$	$\mathcal{P}, \emptyset$	$\mathcal{P}, \succ, 1$	$\mathcal{P}, \succ, 3$	$\mathcal{P}, \succ, 5$
q_1 [5]	#	910	910	910	910	910
	time	14435	13269	12107	13726	12895
q_2 [4]	#	1555	252	252	521	521
	time	12105	12570	11835	13925	14149
q_3 [6]	#	17254	14797	17254	17254	17254
	time	13023	13182	14365	13592	15446
q_4 [5]	#	1301	332	332	1071	1071
	time	12713	11337	12556	14058	14161
q_5 [10]	#	96657	13028	96657	96657	96657
	time	22619	15572	22884	23973	25540
q_6 [7]	#	1446	612	1446	1446	1446
	time	14180	17642	15606	17045	16116
q_7 [5]	#	5267	4431	5267	5267	5267
	time	25524	38315	50749	65764	67492
q_8 [9]	#	339	311	311	311	311
	time	13208	14616	13061	14391	14370
q_9 [6]	#	477	47	47	448	448
	time	12209	13848	10986	14202	14406

Table 6.2. k-DD censor results for the five considered settings. $\emptyset, \emptyset$: empty policy and empty priority relation; $\mathcal{P}, \emptyset$: policy $\mathcal{P}$ and empty priority relation; $\mathcal{P}, \succ, i$: policy $\mathcal{P}$, priority relation $\succ$, and $k = i$, with $i \in \{1, 3, 5\}$.

$\mathcal{P}$, generate challenging scenarios for our technique.

$$\begin{aligned}
\mathsf{wellboreType} &\succ \mathsf{dateWellboreEntry},\\
\mathsf{coreForWellbore} &\succ \mathsf{documentForWellbore},\\
\mathsf{licenseeForLicence} &\succ \mathsf{currentFieldOwner},\\
\mathsf{wellboreCompletionYear} &\succ \mathsf{documentForWellbore},\\
\mathsf{wellboreCompletionYear} &\succ \mathsf{wellOperator},\\
\mathsf{oilSampleTestForWellbore} &\succ \mathsf{wellboreCompletionYear}.
\end{aligned}$$

For each setting, we computed a new mapping through a Java implementation of the algorithm *PolicyEmbed* illustrated in Section 6.3. We performed the experiments through the Java API of the OBDA Mastro system [61] over a standard laptop with an Intel i7-1065G7 @1.3Ghz processor and 8 GB of RAM.

Table 6.2 reports the results of our experiments. The row "#" for each query q_i displays the number of tuples in its evaluation, whereas the row "time" indicates the evaluation time in milliseconds. Finally, the length of each query, i.e. the number of atoms occurring in it, is indicated in square brackets near the query.

The result sets' sizes reported in the second column show that the policy $\mathcal{P}$ has an effect on query answering for eight of the nine queries (query q_1 is the only one not altered by any censor), often hiding several answers with respect to the setting with no policy (up to a tenth of the starting size). By introducing the priority relation, already with $k = 1$, we recover the whole set of original answers for queries

q_3, q_5, q_6, and q_7. As for $k = 3$, we get a noticeable recovery of original answers for queries q_2, q_4 and q_9. Finally, q_8 never recovers even part of the answers that were censored in the case without priorities, even when we reach $k = 5$.

For measuring the query performance, we ran each query ten times in each setting and averaged the execution times. Additionally, all such executions were carried out independently in a pseudo-randomized order, in order to prevent DBMS caching from affecting the results. With the exception of the seventh query, whose running time increases as the complexity of the considered scenario grows up, we can see that the evaluation times remain quite similar across the various settings. Interestingly, the evaluation time sometimes decreases with respect to the case in which the priority relation is not specified. This is related to the fact that, for each atom α, when we adopt a priority relation, $DD_1(\alpha)$ contains fewer conditions than the case with empty priority relation (i.e. it is syntactically shorter than $\mathit{Discl}(\alpha)$). Our findings suggest that our proposed technique can be effectively employed for protecting data in an OBDA framework.

Chapter 7

Dynamic controlled query evaluation

This chapter addresses the subject of dynamic CQE, i.e. an extension of the framework presented in the previous chapters that deals with responding to the user not only on the basis of the current query, but also according to what she asked in the past. This setting allows us to be *maximally cooperative* with the user, i.e. to respond positively whenever possible. This is a contrast to skeptical reasoning, which in general takes a more cautious approach.

In the first part, we formally model our scenario and the properties which should be satisfied by the new query entailment semantics. Then we focus on the complexity of the entailment problem, proving its FO rewritability and providing some insights on the practical aspects of the proposed technique.

7.1 Dynamic framework and the longest honeymoon property

We want to model a scenario where a user is capable of submitting multiple queries to a CQE system. Such queries are evaluated one after the other, and each such evaluation provides the user with new information. The distinguishing feature of this framework is that the adopted query answering semantics takes specifically into account which information has been acquired by the user thanks to the queries she asked previously. In other terms, differently from what happens in a "static" scenario, we keep trace of all the queries submitted by the user and the order in which they have been asked. As we formalize later, this choice allows our framework to disclose more information to the user while preserving the confidentiality of sensitive information. Intuitively, if a query is entailed by the ontology, then we can answer positively as long as, together with previous answers, this does not contribute to violate the policy. Thus we do not check anymore that the query is true according to all the optimal censors, but only to the ones entailing it along with the previous ones. In this way, the set of suitable censors will be progressively reduced, resulting in a higher throughput of positive answers.

The following notion of *protection state* captures the history of queries submitted by the users to the system.

Definition 25. [State] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification. A *protection state* of $\mathcal{E}$ (or simply *state* of $\mathcal{E}$) is a pair $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$, where $\mathcal{A}$ is an ABox and $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ (with $n \geq 0$) a sequence of Boolean queries.

For convenience, with a little abuse of notation, given any sequence of queries $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$, we also use the subscripted symbol $\mathcal{Q}_i$ for indicating the subsequence consisting of the first i queries $q_1, \ldots, q_i$, with the convention that $\mathcal{Q}_0$ is the empty sequence $\langle\rangle$. Analogously, given any state $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$ we also use the symbol $\mathcal{S}_i$ for indicating the state $\langle \mathcal{E}, \mathcal{A}, \mathcal{Q}_i \rangle$.

Given a CQE specification $\mathcal{E}$, an ABox $\mathcal{A}$ and a Boolean query q, we denote by $\mathit{optGACens}(\mathcal{E}, \mathcal{A}, q)$ the following subset of $\mathit{optGACens}(\mathcal{E})$:

$$\mathit{optGACens}(\mathcal{E}, \mathcal{A}, q) = \{\mathit{cens}(\cdot) \in \mathit{optGACens}(\mathcal{E}) \mid \mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q\}$$

Below we formalize our idea of dynamic CQE, i.e. a CQE that takes into account the sequence of queries that have been already processed.

Definition 26. [dynCQE semantics] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, and let $\mathcal{Q}_n = \langle q_1, \ldots, q_n \rangle$ (with $n \geq 0$) a sequence of Boolean queries. The set $\mathit{stCens}(\mathcal{S}_n)$ of censors of $\mathcal{S}_n$ is inductively defined as follows:

- $\mathit{stCens}(\mathcal{S}_0) = \mathit{optGACens}(\mathcal{E})$;
- $\mathit{stCens}(\mathcal{S}_{i+1}) = \begin{cases} \mathit{stCens}(\mathcal{S}_i) & \text{if } \mathit{stCens}(\mathcal{S}_i) \cap \mathit{optGACens}(\mathcal{E}, \mathcal{A}, q_{i+1}) = \emptyset, \\ \mathit{stCens}(\mathcal{S}_i) \cap \mathit{optGACens}(\mathcal{E}, \mathcal{A}, q_{i+1}) & \text{otherwise} \end{cases}$
  for every $0 \leq i \leq n-1$.

For each query q_i occurring in $\mathcal{Q}_n$, we say that q_i *is entailed by* $\mathcal{S}_n$, denoted by $\mathcal{S}_n \models q_i$, if $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q_i$ for every $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S}_n)$. We denote by $\mathit{EntQ}(\mathcal{S}_n)$ the set of queries occurring in $\mathcal{Q}_n$ that are entailed by $\mathcal{S}_n$, i.e.:

$$\mathit{EntQ}(\mathcal{S}_n) = \{q \in \mathcal{Q}_n \mid \mathcal{S}_n \models q\}.$$

Informally speaking, each set $\mathit{stCens}(\mathcal{S}_i)$ (with $1 \leq i \leq n$) in Definition 26 progressively selects the set of optimal GA censors for $\mathcal{E}$ that agree with $\mathit{EntQ}(\mathcal{S}_i)$. Clearly, if none of the surviving censors $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S}_i)$ is such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q_{i+1}$, then $q_{i+1} \notin \mathit{EntQ}(\mathcal{S}_{i+1})$, so we have that $\mathit{stCens}(\mathcal{S}_{i+1}) = \mathit{stCens}(\mathcal{S}_i)$. On the other hand, if $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q_{i+1}$ for at least one of the surviving censors in $\mathit{stCens}(\mathcal{S}_i)$, then, according to the dynCQE semantics, we have a positive answer, and $\mathit{stCens}(\mathcal{S}_{i+1})$ keeps only the censors in $\mathit{stCens}(\mathcal{S}_i)$ that agree with this answer.

As a result, the stream of queries is processed greedily, answering the truth as long as some of the censors in $\mathit{stCens}(\mathcal{S}_n)$ allow to do it (*longest honeymoon approach* [32]), as we formally show below.

Note that, by Definition 26, given a state $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$ and a query q occurring in $\mathcal{Q}$, we have that either $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$ for every $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S})$, or $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \not\models q$ for every $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S})$. This means that $\mathcal{S} \models q$ iff there exists a censor $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S})$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$.

All such properties are formally enounced in the next lemma.

Lemma 8. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ (with $n \geq 1$) be a sequence of Boolean queries, and let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$. Then we have that, for every i and j such that $0 \leq i \leq j \leq n$,*

(i) $\mathit{stCens}(\mathcal{S}_i) \supseteq \mathit{stCens}(\mathcal{S}_j) \supset \emptyset$;

(ii) *for every* $\mathit{cens}_i(\cdot) \in \mathit{stCens}(\mathcal{S}_i)$ *and* $\mathit{cens}_j(\cdot) \in \mathit{stCens}(\mathcal{S}_j)$, $\mathcal{T} \cup \mathit{cens}_i(\mathcal{A}) \models q_i$ *iff* $\mathcal{T} \cup \mathit{cens}_j(\mathcal{A}) \models q_i$;

(iii) $\mathit{EntQ}(\mathcal{S}_i) \subseteq \mathit{EntQ}(\mathcal{S}_j)$.

Proof. The first property is a direct consequence of Definition 26 (it easy to see that the set of censors of a state is always non-empty).

Furthermore, we know that either every censor $\mathit{cens}_i(\cdot) \in \mathit{stCens}(\mathcal{S}_i)$ is such that q_i evaluates to true in $\mathit{cens}_i(\mathcal{A})$ or none of them is. As a result, the second property holds when $i = j$, and it also holds when $i < j$ due to the inclusion $\mathit{stCens}(\mathcal{S}_i) \supseteq \mathit{stCens}(\mathcal{S}_j)$.

Finally, the third property is a direct consequence of (the *only-if* direction of) the second one. ∎

Example 34. Some pharmaceutical products may reveal with high accuracy which kind of disease is affecting a person. For instance, drugs containing phenytoin as active ingredient, or that are classified as anti-seizure medications, indicate some form of epilepsy.

Let a TBox $\mathcal{T}$, a policy $\mathcal{P}$ and an ABox $\mathcal{A}$ be as follows:

$$\mathcal{T} = \{\mathsf{BrandX} \sqsubseteq \mathsf{Antiseizure}\};$$
$$\mathcal{P} = \{\exists x, y\,(\mathsf{buy}(x, y) \wedge \mathsf{Antiseizure}(y)) \rightarrow \bot,$$
$$\exists x, y\,(\mathsf{buy}(x, y) \wedge \mathsf{contain}(y, \mathsf{phenytoin})) \rightarrow \bot\};$$
$$\mathcal{A} = \{\mathsf{buy}(\mathsf{john}, \mathsf{m}_a), \mathsf{BrandX}(\mathsf{m}_a), \mathsf{buy}(\mathsf{alice}, \mathsf{m}_b), \mathsf{contain}(\mathsf{m}_b, \mathsf{phenytoin})\}.$$

In words, the TBox states that BrandX is an anti-seizure medication, while the policy conceals the presence of patients suffering from epilepsy.

Let us start by considering an empty sequence of queries. By definition, we have that $\mathit{stCens}(\mathcal{S}_0)$ coincides with the set of the optimal GA censors for $\mathcal{E}$. Despite there exists an infinite number of optimal GA censors for $\mathcal{E}$, one can see that, for every $\mathit{cens}(\cdot) \in \mathit{optGACens}(\mathcal{E})$, $\mathit{cens}(\mathcal{A})$ must be one of the following four ABoxes:

$$\mathcal{C}_1 = \{\mathsf{buy}(\mathsf{john}, \mathsf{m}_a), \mathsf{buy}(\mathsf{alice}, \mathsf{m}_b)\};$$
$$\mathcal{C}_2 = \{\mathsf{buy}(\mathsf{john}, \mathsf{m}_a), \mathsf{contain}(\mathsf{m}_b, \mathsf{phenytoin})\};$$
$$\mathcal{C}_3 = \{\mathsf{BrandX}(\mathsf{m}_a), \mathsf{Antiseizure}(\mathsf{m}_a), \mathsf{buy}(\mathsf{alice}, \mathsf{m}_b)\};$$
$$\mathcal{C}_4 = \{\mathsf{BrandX}(\mathsf{m}_a), \mathsf{Antiseizure}(\mathsf{m}_a), \mathsf{contain}(\mathsf{m}_b, \mathsf{phenytoin})\}.$$

Let $q_1 = \mathsf{buy}(\mathsf{john}, \mathsf{m}_a)$ be the first query. Among the above ABoxes, we have that $\mathcal{C}_1$ and $\mathcal{C}_2$ are the only one that, together with the TBox, entail q_1. All the optimal GA censors for $\mathcal{E}$ that disagree with such an answer are then discarded, obtaining that, for each $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S}_1)$, either $\mathit{cens}(\mathcal{A}) = \mathcal{C}_1$ or $\mathit{cens}(\mathcal{A}) = \mathcal{C}_2$.

Now, let $q_2 = \mathsf{BrandX}(\mathsf{m}_a)$ be a new query in the sequence. Since neither $\mathcal{T} \cup \mathcal{C}_1$ nor $\mathcal{T} \cup \mathcal{C}_2$ entail q_2, then $stCens(\mathcal{S}_2) = stCens(\mathcal{S}_1)$. Finally, consider to add $q_3 = \exists x\, \mathsf{buy}(x, \mathsf{m}_b)$ to the sequence. Since $\mathcal{T} \cup \mathcal{C}_1 \models q_3$ while $\mathcal{T} \cup \mathcal{C}_2 \not\models q_3$, we have that $stCens(\mathcal{S}_3)$ contains only the censors $cens(\cdot) \in stCens(\mathcal{S}_2)$ such that $cens(\mathcal{A}) = \mathcal{C}_1$. Clearly, $\mathcal{S}_3 \models q_1$ and $\mathcal{S}_3 \models q_3$, but $\mathcal{S}_3 \not\models q_2$. ▽

Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification. For every state $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$, our dynamic CQE method is *optimal with respect to* $\mathcal{S}$, in the sense that $EntQ(\mathcal{S})$ is never strictly contained in the set of queries of $\mathcal{Q}$ entailed by $cens(\mathcal{A})$, taken any GA censor $cens(\cdot)$ for $\mathcal{E}$.

Proposition 17. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ be a sequence of Boolean queries, and let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$. There exists no GA censor $cens(\cdot) \in optGACens(\mathcal{E})$ such that $EntQ(\mathcal{S}) \subset \{q \in \mathcal{Q} \mid \mathcal{T} \cup cens(\mathcal{A}) \models q\}$.*

Proof. By contradiction, let such a GA censor $cens(\cdot)$ exist and let i be the least index such that $\mathcal{T} \cup cens(\mathcal{A}) \models q_i$ and $q_i \notin EntQ(\mathcal{S})$. By the minimality of i we have that, for every $j \in \{1, \ldots, i-1\}$, $\mathcal{T} \cup cens(\mathcal{A}) \models q_j$ iff $q_j \in EntQ(\mathcal{S}_{i-1})$. It follows that $cens(\cdot) \in stCens(\mathcal{S}_{i-1})$. But then, by definition, we should have that $cens(\cdot) \in stCens(\mathcal{S}_i)$ and, consequently, that $q_i \in EntQ(\mathcal{S}_i) \subseteq EntQ(\mathcal{S})$ (a contradiction). ∎

Moreover, dynCQE is the *only* semantics that guarantees the preservation of such optimality in the future. One might object that answering the current query q honestly may prevent the system from answering honestly another set of queries $\mathcal{Q}'$ in the future. However, the queries in $\mathcal{Q}'$ might never be submitted, so any GA censor that conceals the answer to q now might remain sub-optimal in the future. This may happen no matter how many additional queries are submitted by the users. Formally, we have:

Proposition 18. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ be a sequence of Boolean queries, and let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$. For every query q_{n+1} and for every censor $cens(\cdot) \in stCens(\mathcal{S}) \setminus stCens(\langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \circ \langle q_{n+1} \rangle \rangle)$[1], there exists a sequence of queries $q_{n+2}, q_{n+3}, \ldots, q_{n+k}, \ldots$ such that, for every $i > n$:*

$$\{q \in \langle q_1, \ldots, q_i \rangle \mid \mathcal{T} \cup cens(\mathcal{A}) \models q\} \subset EntQ(\langle \mathcal{E}, \mathcal{A}, \langle q_1, \ldots, q_i \rangle \rangle).$$

Proof (sketch). It follows from the hypothesis that $\mathcal{T} \cup cens(\mathcal{A}) \not\models q_{n+1}$, therefore:

$$\begin{aligned} \{q \in \langle q_1, \ldots, q_i \rangle \mid \mathcal{T} \cup cens(\mathcal{A}) \models q\} &= EntQ(\langle \mathcal{E}, \mathcal{A}, \langle q_1, \ldots, q_{n+1} \rangle \rangle) \setminus \{q_{n+1}\} \\ &\subset EntQ(\langle \mathcal{E}, \mathcal{A}, \langle q_1, \ldots, q_{n+1} \rangle \rangle) \end{aligned}$$

The simplest case in which this strict inclusion is preserved for all the future queries is where, for every positive integer k, the query q_{n+1+k} is picked from $\{q_1, \ldots, q_{n+1}\}$ (or logically equivalent formulae thereof). In this way, for every $k > 0$, we have that $\{q \in \langle q_1, \ldots, q_{n+1+k} \rangle \mid \mathcal{T} \cup cens(\mathcal{A}) \models q\} = \{q \in \langle q_1, \ldots, q_{n+1} \rangle \mid \mathcal{T} \cup cens(\mathcal{A}) \models q\}$, and $EntQ(\langle \mathcal{E}, \mathcal{A}, \langle q_1, \ldots, q_{n+1+k} \rangle \rangle) = EntQ(\langle \mathcal{E}, \mathcal{A} \langle q_1, \ldots, q_{n+1} \rangle \rangle)$. ∎

[1] With $\mathcal{Q} \circ \langle q_{n+1} \rangle$ we denote the sequence $\langle q_1, \ldots, q_n, q_{n+1} \rangle$.

In the above proposition, the hypothesis $cens(\cdot) \in stCens(\mathcal{S}) \setminus stCens(\langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \circ \langle q_{n+1} \rangle \rangle)$ implies that q_{n+1} can be given a positive answer without disclosing any protected data, but $cens(\mathcal{A})$ does not allow a positive answer to q_{n+1}.

In practical applications, queries are typically submitted sequentially, and the system is unaware of which queries, if any, follow at each step. In this scenario, it is reasonable to adopt the so-called *longest honeymoon approach*: "one must keep telling the whole truth as long as it is possible". The following notion of maximally cooperative censor formally captures the idea of longest honeymoon.

Definition 27. [Cooperativity] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ (with $n \geq 0$) a sequence of Boolean queries, and let $cens(\cdot)$ and $cens'(\cdot)$ two optimal GA censors for $\mathcal{E}$. We say that $cens(\cdot)$ is *more cooperative than* $cens'(\cdot)$ with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$ if there exists a non-negative natural number $m < n$ such that:

- $\mathcal{T} \cup cens(\mathcal{A}) \models q_i \iff \mathcal{T} \cup cens'(\mathcal{A}) \models q_i$ for every $1 \leq i \leq m$, and
- $\mathcal{T} \cup cens(\mathcal{A}) \models q_{m+1}$ and $\mathcal{T} \cup cens'(\mathcal{A}) \not\models q_{m+1}$.

We also say that $cens(\cdot)$ is *maximally cooperative* with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$ if there does not exist any GA censor $cens''(\cdot)$ for $\mathcal{E}$ that is more cooperative than $cens(\cdot)$ with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$.

Example 35. Consider the CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, the ABox $\mathcal{A}$, the sequence of queries $\mathcal{Q} = \langle q_1, q_2, q_3 \rangle$, and the GA censors for $\mathcal{E}$ as defined in Example 34. It is straightforward to see that all the optimal GA censors $cens(\cdot)$ such that $cens(\mathcal{A}) = \mathcal{C}_1$ are all (and the only ones) *maximally cooperative* with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$. ▽

The following intermediate result shows that a state of a CQE specification cannot discriminate between two optimal GA censors if they have answered all the queries posed so far in the same way.

Lemma 9. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ (with $n \geq 0$) be a sequence of Boolean queries, let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$, and let $cens(\cdot)$ and $cens'(\cdot)$ be two optimal GA censors for $\mathcal{E}$ such that $\mathcal{T} \cup cens(\mathcal{A}) \models q_i \iff \mathcal{T} \cup cens'(\mathcal{A}) \models q_i$, for every $i \in \{1, \ldots, n\}$. Then, $cens(\cdot) \in stCens(\mathcal{S})$ iff $cens'(\cdot) \in stCens(\mathcal{S})$.*

Proof. The proof is by induction on the length of $\mathcal{Q}$.

◊ Case $n = 0$. Since $\mathcal{Q}$ is empty, both $cens(\cdot)$ and $cens'(\cdot)$ are in $stCens(\mathcal{S})$.

◊ Case $n \geq 1$. From the assumption $\mathcal{T} \cup cens(\mathcal{A}) \models q_i$ iff $\mathcal{T} \cup cens'(\mathcal{A}) \models q_i$, for every $i \in \{1, \ldots, n\}$, the following two facts hold:

(*i*) $cens(\cdot) \in optGACens(\mathcal{E}, \mathcal{A}, q_n)$ iff $cens'(\cdot) \in optGACens(\mathcal{E}, \mathcal{A}, q_n)$;

(*ii*) by I.H., $cens(\cdot) \in stCens(\mathcal{S}_{n-1})$ iff $cens'(\cdot) \in stCens(\mathcal{S}_{n-1})$.

Then, since $stCens(\mathcal{S})$ is by Definition 26 equal either to $stCens(\mathcal{S}_{n-1})$ or to $stCens(\mathcal{S}_{n-1}) \cap optGACens(\mathcal{E}, \mathcal{A}, q_n)$, we have the thesis.

■

Now, in Example 35 we have that the maximally cooperative censors with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$ corresponds precisely to those selected by the dynCQE semantics. This is not a coincidence; in fact we can show that, for any state $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$ of a CQE specification, the set $stCens(\mathcal{S})$ coincides with the set of all censors that are maximally cooperative with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$.

Theorem 13. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ (with $n \geq 0$) be a sequence of Boolean queries, and let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$. An optimal GA censor $cens(\cdot)$ for $\mathcal{E}$ is maximally cooperative with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$ iff $cens(\cdot) \in stCens(\mathcal{S})$.*

Proof. We start by showing that if an optimal GA censor $cens(\cdot)$ for $\mathcal{E}$ is maximally cooperative with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$, then $cens(\cdot) \in stCens(\mathcal{S})$. By contradiction, assume that $cens(\cdot) \notin stCens(\mathcal{S})$. So, there exists an index $i > 0$ of some query $q_i \in \mathcal{Q}$ such that $cens(\cdot) \in stCens(\mathcal{S}_{i-1}) \setminus stCens(\mathcal{S}_i)$. Hence, there exists a censor $cens'(\cdot) \in stCens(\mathcal{S}_i)$ such that $\mathcal{T} \cup cens'(\mathcal{A}) \models q_i$, while $\mathcal{T} \cup cens(\mathcal{A}) \not\models q_i$ and such that $\mathcal{T} \cup cens'(\mathcal{A}) \models q_j \iff \mathcal{T} \cup cens(\mathcal{A}) \models q_j$ for every $1 \leq j \leq i-1$. So, by Definition 27, $cens'(\cdot)$ is more cooperative than $cens(\cdot)$, which contradicts the fact that $cens(\cdot)$ is maximally cooperative with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$.

Now, we show that every $cens(\cdot) \in stCens(\mathcal{S})$ is maximally cooperative with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$. Assume by contradiction that, for some $cens(\cdot) \in stCens(\mathcal{S}_n)$, there exists a more cooperative censor $cens'(\cdot)$. This means that, for some integer m such that $0 \leq m < n$, (i) $\mathcal{T} \cup cens(\mathcal{A}) \models q_i \iff \mathcal{T} \cup cens'(\mathcal{A}) \models q_i$, for each $i \leq m$, and (ii) $\mathcal{T} \cup cens(\mathcal{A}) \not\models q_{m+1}$ and $\mathcal{T} \cup cens'(\mathcal{A}) \models q_{m+1}$.

From the assumption $cens(\cdot) \in stCens(\mathcal{S}_n)$ and by Lemma 8, we have that $cens(\cdot)$ occurs also in $stCens(\mathcal{S}_m)$ and $stCens(\mathcal{S}_{m+1})$. From $cens(\cdot) \in stCens(\mathcal{S}_m)$, condition (i) and Lemma 9, we have that $cens'(\cdot)$ belongs to $stCens(\mathcal{S}_m)$ too.

From condition (ii) we have that $cens'(\cdot)$ occurs in $optGACens(\mathcal{E}, \mathcal{A}, q_{m+1})$ whereas $cens(\cdot)$ does not. Now, since $cens'(\cdot) \in stCens(\mathcal{S}_m) \cap optGACens(\mathcal{E}, \mathcal{A}, q_{m+1})$, then $stCens(\mathcal{S}_{m+1})$ is equal by definition to $stCens(\mathcal{S}_m) \cap optGACens(\mathcal{E}, \mathcal{A}, q_{m+1})$. Anyway, since $cens(\cdot)$ is not in $optGACens(\mathcal{E}, \mathcal{A}, q_{m+1})$, we have that $stCens(\mathcal{S}_{m+1})$ does not contain $cens(\cdot)$, which means that also $stCens(\mathcal{S}_n)$ does not contain $cens(\cdot)$, leading to a contradiction. ■

In the following, we focus on the decision problem dynCQE-Ent, which we define below.

Definition 28. [dynCQE-Ent] Given as input a CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, an ABox $\mathcal{A}$, a non-empty sequence $\mathcal{Q}$ of Boolean queries, and a query $q \in \mathcal{Q}$, dynCQE-Ent is the problem of deciding whether $q \in EntQ(\mathcal{S})$, where $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$.

We conclude this section by comparing dynCQE to the other semantics already discussed in this thesis. First consider the strategy consisting in arbitrarily choosing an optimal GA censor (already mentioned in Section 3.2). In this case it might happen, as also stated by Proposition 18, that one looses optimality with respect to the state $\mathcal{S}$. For instance, if in Example 34 one arbitrarily picks a GA censor $cens(\cdot)$ such that $cens(\mathcal{A}) = \mathcal{C}_2$, then $\{q \in \mathcal{Q} \mid \mathcal{T} \cup cens(\mathcal{A}) \models q\} \subset EntQ(\mathcal{S})$. The

following proposition shows that, given any CQE specification, this approach may always lead to going against the longest-honeymoon approach.

Proposition 19. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, and let $cens(\cdot)$ be an optimal GA censor for $\mathcal{E}$. If there exists an optimal GA censor $cens'(\cdot)$ for $\mathcal{E}$ such that $cens(\mathcal{A}) \neq cens'(\mathcal{A})$, then there exists a sequence $\mathcal{Q}$ for which $cens(\cdot)$ is not maximally cooperative with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$.*

Proof. From $cens(\mathcal{A}) \neq cens'(\mathcal{A})$ we have that either $cens(\mathcal{A}) \supset cens'(\mathcal{A})$ or $cens(\mathcal{A}) \not\supseteq cens'(\mathcal{A})$. Anyway, one can see that the first case would result in a contradiction. Indeed, consider the GA censor $cens''(\cdot)$ corresponding to $cens'(\cdot)$ for each input ABox but $\mathcal{A}$, for which we have instead $cens''(\mathcal{A}) = cens(\mathcal{A})$. Clearly, if $cens(\mathcal{A}) \supset cens'(\mathcal{A})$, then $cens''(\cdot)$ would be more informative than $cens'(\cdot)$, which in turn would not be optimal, thus contradicting the hypothesis.

Since it must hold $cens(\mathcal{A}) \not\supseteq cens(\mathcal{A})$, then there is at least one ground atom $\alpha \in cens'(\mathcal{A})$ that is not contained in $cens(\mathcal{A})$. Now, consider the sequence $\mathcal{Q} = \langle q \rangle$, where $q = \alpha$. Since $\mathcal{T} \cup cens(\mathcal{A}) \not\models q$ and $\mathcal{T} \cup cens'(\mathcal{A}) \models q$, then $cens'(\cdot)$ is more cooperative than $cens(\cdot)$ with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$. Thus, $cens(\cdot)$ is not maximally cooperative with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$. ∎

As for the case of prioritized CQE, for each semantics presented in Chapter 4 an analogous of Proposition 19 can be formulated. Intuitively, for some prioritized CQE specification $\mathcal{E}_\succ$ and some ABox $\mathcal{A}$, the priority relation may induce the entailment semantics to "ignore" all the censors $cens'(\cdot)$ for $\mathcal{E}_\succ$ such that $cens'(\mathcal{A})$ contains a certain fact α. In this case, it can be proven in a similar way as above that there always exists a sequence $\mathcal{Q}$ such that every survived censor is not maximally cooperative with respect to $\langle \mathcal{Q}, \mathcal{A} \rangle$.

For what concerns skeptically reasoning under all optimal GA censors, we show that such a semantics is always a sound approximation of dynCQE.

Proposition 20. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ be a non-empty sequence of Boolean queries, let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$, and let $q \in \mathcal{Q}$. If $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{GA}} q$ holds, then $\mathcal{S} \models q$ holds too. The converse is not necessarily true.*

Proof. By definition, if $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{GA}} q$ then $\mathcal{T} \cup cens(\mathcal{A}) \models q$ holds for each $cens(\cdot) \subseteq optGACens(\mathcal{E})$. In particular, it holds for every $cens(\cdot) \in stCens(\mathcal{S})$, and therefore we have that $\mathcal{S} \models q$.

As for the converse, consider Example 34. We have that $\mathcal{S} \models q_1$, but $\langle \mathcal{E}, \mathcal{A} \rangle \not\models_{\mathsf{GA}} q_1$ because $\mathcal{T} \cup \mathcal{C}_3 \not\models q_1$. ∎

Then, the following statement is an immediate consequence of Proposition 7 and Proposition 20.

Corollary 4. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ be a non-empty sequence of Boolean queries, let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$, and let $q \in \mathcal{Q}$. If $\langle \mathcal{E}, \mathcal{A} \rangle \models_{\mathsf{IGA}} q$ holds, then $\mathcal{S} \models q$ holds too.*

7.2 First-order rewritability of BUCQ entailment

We now move to the study of the computational complexity of query entailment in the dynamic CQE framework. As for previous complexity analyses, in the rest of this chapter we examine the case of DL-Lite$_\mathcal{R}$ CQE specifications and queries expressed as BUCQs.

For convenience, in the rest of this section, we overload the function *PerfectRef*$(\cdot,\cdot)$ by making it accept as first argument a policy $\mathcal{P}$, i.e. we use *PerfectRef*$(\mathcal{P},\mathcal{T})$ as an abbreviation for *PerfectRef*$(\bigvee_{q\to\bot\in\mathcal{P}} q,\mathcal{T})$. We then rephrase Proposition 16 as follows:

Proposition 21. *Let $\mathcal{T}\cup\mathcal{A}$ be a consistent DL-Lite$_\mathcal{R}$ ontology, and let $\mathcal{P}$ be a policy. Then, $\mathcal{T}\cup\mathcal{P}\cup\mathcal{A}$ is a consistent FO theory iff PerfectRef$(\mathcal{P},\mathcal{T})$ evaluates to false in $\mathcal{A}$.*

A first way to solve query entailment in a state might consist in finding a reduction to the stateless CQE approach, for which algorithms are already known. It turns out, however, that the behavior of the dynCQE semantics cannot be intensionally simulated through a stateless CQE specification independent of query history.

Theorem 14. *There exist a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}=\langle\mathcal{T},\mathcal{P}\rangle$ and a BUCQ q such that there exist no DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}'=\langle\mathcal{T}',\mathcal{P}'\rangle$ such that, for every ABox $\mathcal{A}$, optGACens$(\mathcal{E}')=$ stCens$(\mathcal{S})$, where $\mathcal{S}=\langle\mathcal{E},\mathcal{A},\langle q\rangle\rangle$.*

Proof. Let $\mathcal{T}=\emptyset$, let $\mathcal{P}=\{\forall x\,(C(x)\wedge D(x)\to\bot)\}$, and let $q=\exists x\,C(x)$. By contradiction, suppose there exists a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E}'=\langle\mathcal{T}',\mathcal{P}'\rangle$ such that, for every ABox $\mathcal{A}$, *optGACens*$(\mathcal{E}')=$ *stCens*$(\mathcal{S})$.

Now consider the ABox $\mathcal{A}=\{C(a_1),C(a_2),D(a_1),D(a_2)\}$, where a_1,a_2 are individual names that do not appear in $\mathcal{P}'$. Every optimal censor *cens*$(\cdot)$ for $\mathcal{E}=\langle\mathcal{T},\mathcal{P}\rangle$ is such that *cens*$(\mathcal{A})$ is equal to one of the following ABoxes: $\mathcal{C}_1=\{C(a_1),C(a_2)\}$, $\mathcal{C}_2=\{C(a_1),D(a_2)\}$, $\mathcal{C}_3=\{D(a_1),C(a_2)\}$, $\mathcal{C}_4=\{D(a_1),D(a_2)\}$. Among such ABoxes, $\mathcal{C}_4$ is the only one for which $\mathcal{T}\cup\mathcal{C}_4\not\models q$. Therefore, given $\mathcal{A}$ as input, every censor of *stCens*$(\mathcal{S})$ returns either $\mathcal{C}_1$, $\mathcal{C}_2$ or $\mathcal{C}_3$. Since by hypothesis *stCens*$(\mathcal{S})=$ *optGACens*$(\mathcal{E}')$, it follows that:

- $\mathcal{T}'\cup\mathcal{P}'\cup\mathcal{C}_3$ is consistent;
- $\mathcal{T}'\cup\mathcal{P}'\cup\mathcal{C}_4$ is inconsistent. Indeed, since $\mathcal{C}_4\subseteq\mathcal{A}$, if $\mathcal{T}'\cup\mathcal{P}'\cup\mathcal{C}_4$ was consistent then $\mathcal{C}_4$ would be contained in *cens*$(\mathcal{A})$ for some *cens*$(\cdot)\in$ *optGACens*$(\mathcal{E}')$, i.e. it would be contained in some $\mathcal{C}_i$, with $1\le i\le 3$.

Consequently, by Proposition 21, *PerfectRef*$(\mathcal{P}',\mathcal{T}')$ evaluates to true in $\mathcal{C}_4$ and it evaluates to false in $\mathcal{C}_3$.

On the other hand, it is immediate to see that, for every BUCQ q that does not mention individual names in $\mathcal{A}$, the query q evaluates to true in $\mathcal{C}_4$ only if q evaluates to true in $\mathcal{C}_3$. Consequently, *PerfectRef*$(\mathcal{P}',\mathcal{T}')$ evaluates to true in $\mathcal{C}_4$ only if *PerfectRef*$(\mathcal{P}',\mathcal{T}')$ evaluates to true in $\mathcal{C}_3$. Thus we get a contradiction. ∎

We now shall prove that dynCQE-Ent is FO rewritable, and therefore in AC^0 in data complexity. We start by showing a fundamental property of query entailment in a state, which holds for every DL.

Proposition 22. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ be a sequence of Boolean queries, and let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$. For every i such that $1 \leq i \leq n$, $q_i \in \mathit{EntQ}(\mathcal{S})$ iff there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that*

$$\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models \Big(\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_{i-1})} q \Big) \wedge q_i.$$

Proof. ($\Leftarrow$:) Suppose there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models (\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_{i-1})} q) \wedge q_i$. By Definition 26, we immediately have that $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S}_i)$ and $q_i \in \mathit{EntQ}(\mathcal{S}_i)$, which, by the third point of Lemma 8, implies that $q_i \in \mathit{EntQ}(\mathcal{S})$.

($\Rightarrow$:) Suppose now q_i is a query belonging to $\mathit{EntQ}(\mathcal{S})$, and consider a censor $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S})$ (which always exists). Then, we have that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$ for every $q \in \mathit{EntQ}(\mathcal{S})$. In particular, this holds for q_i and for every $q \in \mathit{EntQ}(\mathcal{S}_{i-1})$ (since, by Lemma 8, $\mathit{EntQ}(\mathcal{S}_{i-1}) \subseteq \mathit{EntQ}(\mathcal{S})$). Therefore, we have that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models (\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_{i-1})} q) \wedge q_i$, which proves the thesis. ∎

We are now ready to prove the next result.

Proposition 23. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{R}$ CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ (with $n \geq 0$) be a sequence of BUCQs, and let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$. For every i such that $1 \leq i \leq n$, $q_i \in \mathit{EntQ}(\mathcal{S})$ iff there exists an image M of $\mathit{PerfectRef}((\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_{i-1})} q) \wedge q_i, \mathcal{T})$ in $\mathit{cl}^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ such that $\mathit{PerfectRef}(\mathcal{P}, \mathcal{T})$ evaluates to false in M.*

Proof. First note that, from the definition of GA censor for a CQE specification it straightforwardly follows that there exists a GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$ iff there exists a set of facts $\mathcal{F} \subseteq \mathit{cl}^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ such that $\mathcal{T} \cup \mathcal{F} \models q$ and $\mathcal{T} \cup \mathcal{P} \cup \mathcal{F}$ is consistent. One can also assume w.l.o.g. that $\mathit{cens}(\cdot)$ is optimal for $\mathcal{E}$ (otherwise there would exist a censor $\mathit{cens}'(\cdot)$ such that $\mathit{cens}(\mathcal{A}) \subseteq \mathit{cens}'(\mathcal{A})$ for every ABox $\mathcal{A}$, which would clearly satisfy the same property).

As a consequence Proposition 1, Proposition 21 and what above, there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$ iff there exists an image M of $\mathit{PerfectRef}(q, \mathcal{T})$ in $\mathit{cl}^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ such that $\mathit{PerfectRef}(\mathcal{P}, \mathcal{T})$ evaluates to false in M. Along with this property, Proposition 22 implies the thesis. ∎

Now observe that: (*i*) $\mathit{cl}^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ can be computed in polynomial time with respect to data complexity; (*ii*) the size of every image of a BUCQ q is not larger than the length of the longest BCQ in q; (*iii*) such a maximum length is a constant with respect to data complexity; (*iv*) all the conditions in the theorem can be verified in polynomial time with respect to data complexity [45]. This implies that the entailment problem in a state belongs to PTime with respect to data complexity.

In the following, we provide a tighter upper bound, showing that this entailment problem is in AC^0 in data complexity. We do so by proving that the problem is FO rewritable. That is, for every BUCQ q of the state, there exists an FO query q' that does not depend on the ABox and is such that q is entailed in the state iff q' evaluates to true in the ABox.

To this purpose, we find an FO query that only depends on the intensional part of the state (i.e. the CQE specification and the sequence of queries), and such that

its evaluation on the ABox is true iff the condition expressed in Proposition 23 holds (Theorem 15). Toward this result, we make the following intermediate step: given a query q on a DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, we find a query whose evaluation on $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ corresponds to checking the existence of an optimal GA censor $cens(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup cens(\mathcal{A}) \models q$, for every possible ABox $\mathcal{A}$ (Proposition 24).

Given two BCQs q and q', a *mapping of q' into q* is a total function $h : predAt(q') \to predAt(q)$ such that there exists an MGU σ_h for α and $h(\alpha)$ replacing variables occurring either in q' or in q with either variables of q or constants. We denote by $Map(q', q)$ the set of all the mappings of q' into q. Furthermore, we denote by $\sigma_h[q]$ the variable substitutions of σ_h limited to variables occurring in q. For instance, if $q = \exists x, y, z\, R(x, y, z)$, $q' = \exists x'\, R(x', x', a)$ (where a is a constant and all other arguments are variables), then $\sigma_h = \{x' \mapsto x, y \mapsto x, z \mapsto a\}$ and $\sigma_h[q] = \{y \mapsto x, z \mapsto a\}$.

Given two BCQs q and q', we define the formula $Unify(q, q')$ as follows:

$$Unify(q, q') = \bigvee_{h \in Map(q', q)} \Big(\bigwedge_{x \mapsto t \in \sigma_h[q]} x = t \Big)$$

The following lemma shows a crucial property enjoyed by the function *Unify*.

Lemma 10. *Let $q = \exists \mathbf{x}\, \gamma(\mathbf{x})$ be a BCQ, let q' be a BUCQ, and let $\mathcal{A}$ be an ABox. Then, there exists an image M of q in $\mathcal{A}$ such that q' evaluates to false in M iff the following sentence evaluates to true in $\mathcal{A}$:*

$$\exists \mathbf{x} \Big(\gamma(\mathbf{x}) \wedge \bigwedge_{q_d \in CQ(q')} \neg Unify(q_d, q) \Big).$$

Proof. We show that, if $q' = \exists \mathbf{y}\, \psi(\mathbf{y})$ is a BCQ, then there exists an image M of q in $\mathcal{A}$ such that q' evaluates to false in M iff $\exists \mathbf{x}\, (\gamma(\mathbf{x}) \wedge \neg Unify(q, q'))$ evaluates to true in $\mathcal{A}$. The extension to BUCQs is a direct consequence. Moreover, we assume w.l.o.g. that $\mathbf{x}$ and $\mathbf{y}$ are disjoint.

($\Rightarrow$:) Let M be an image of q in $\mathcal{A}$ such that q' evaluates to false in M, then there exists an instantiation σ_M of the variables in $\mathbf{x}$ such that $M = predAt(\sigma_M(q))$, which also implies that $\sigma_M(\gamma(\mathbf{x}))$ evaluates to true in $\mathcal{A}$. It remains to show that $\sigma_M(Unify(q, q'))$ evaluates to false in $\mathcal{A}$. By contradiction, assume h to be a mapping from q' to q such that $\sigma_M(\bigwedge_{x \mapsto t \in \sigma_h[q]} x = t)$ evaluates to true in $\mathcal{A}$. This implies that σ_M is a specialization of $\sigma_h[q]$, i.e. $\sigma_h[q] \circ \sigma_M = \sigma_M$. Then, for each atom $\beta \in predAt(q')$, we have by construction that:

$$[\sigma_h \circ \sigma_M](\beta) = \sigma_M(\sigma_h(\beta)) = \sigma_M(\sigma_h(h(\beta))) = [\sigma_h[q] \circ \sigma_M](h(\beta)) = \sigma_M(h(\beta)).$$

Since $h(\beta)$ is an atom in q and M is equal to $predAt(\sigma_M(q))$, $[\sigma_h \circ \sigma_M](\beta)$ occurs in M, for each $\beta \in predAt(q')$. But this means that q' evaluates to true in M, which contradicts the hypothesis.

($\Leftarrow$:) $\exists \mathbf{x}\, (\gamma(\mathbf{x}) \wedge \neg Unify(q, q'))$ evaluates to true in $\mathcal{A}$. This means that there exists an answer substitution σ such that (i) $predAt(\sigma(q)) \subseteq \mathcal{A}$, and (ii) $\sigma(Unify(q, q'))$ evaluates to false in $\mathcal{A}$. Let $M = predAt(\sigma(q))$, from (i) we know that M is an image of q in $\mathcal{A}$. Then, assume by contradiction that q' evaluates to true in M. This means that for some answer substitution σ', $predAt(\sigma'(q')) \subseteq M$.

Let h be the mapping of q' into q such that $h(\beta) = \alpha$ only if $\sigma'(\beta) = \sigma(\alpha)$, we know from $\mathit{predAt}(\sigma'(q')) \subseteq M$ that this mapping exists. The corresponding MGU σ_h over $\mathbf{x}$ and $\mathbf{y}$ can be defined as follows.

For the variables y occurring in $\mathbf{y}$, $\sigma_h(y) = t$ where for some $\beta \in \mathit{predAt}(q')$ and $h(\beta) \in \mathit{predAt}(q)$, y and t occurs in the same position, respectively.

For the variables x occurring in $\mathbf{x}$, we have that:

$$\sigma_h(x) = \begin{cases} y & \text{if for some } y \text{ in } \mathbf{y},\ \sigma_h(y) = y \text{ and } \sigma(x) = \sigma(y) \\ \sigma(x) & \text{otherwise.} \end{cases}$$

Note that, for each a variable x in $\mathbf{x}$, either $[\sigma_h[q] \circ \sigma](x) = \sigma(\sigma(x))$ or $[\sigma_h[q] \circ \sigma](x) = \sigma(y)$ with $\sigma(x) = \sigma(y)$. In both the cases, we have that $[\sigma_h[q] \circ \sigma](x) = \sigma(x)$. But this means that σ is a specialization of σ_h, which conflicts with (ii). ∎

We can now define the query reformulation function $\mathit{BraveRef}(\cdot, \cdot)$.

Definition 29. [*BraveRef*] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a CQE specification, and let q be a BUCQ. We define $\mathit{BraveRef}(q, \mathcal{E})$ as the FO sentence:

$$\bigvee_{q_r \in \mathit{PerfectRef}(q, \mathcal{T})} \exists \mathbf{x}_r \left(\gamma_r(\mathbf{x}_r) \wedge \bigwedge_{q_d \in \mathit{PerfectRef}(\mathcal{P}, \mathcal{T})} \neg \mathit{Unify}(q_r, q_d) \right)$$

(where we assume $q_r = \exists \mathbf{x}_r\, \gamma_r(\mathbf{x}_r)$).

Intuitively, the above formula checks if there exists a rewriting q_r of q with respect to $\mathcal{T}$ that evaluates to true for an assignment of its variables under which the atoms of q_r do not imply (the rewriting of) any body of any denial in $\mathcal{P}$. In the following theorem, we show that the evaluation of this formula in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ can be used to decide in a sound and complete way if q is true according to at least one optimal GA censor for $\mathcal{E}$.

Proposition 24. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{R}$ CQE specification. For every ABox $\mathcal{A}$, there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\langle \mathcal{T}, \mathcal{P} \rangle$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$ iff $\mathit{BraveRef}(q, \mathcal{E})$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$.*

Proof. First, from Proposition 1 it follows that, $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q$ iff $\mathit{PerfectRef}(q, \mathcal{T})$ evaluates to true in $\mathit{cens}(\mathcal{A})$, for every optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$.

Now, since $\mathit{cens}(\mathcal{A}) \subseteq cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, and since every subset $\mathcal{A}'$ of $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ such that $\mathcal{T} \cup \mathcal{P} \cup \mathcal{A}'$ is consistent is contained in $\mathit{cens}'(\mathcal{A})$ for some optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$, it follows that there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathit{PerfectRef}(q, \mathcal{T})$ evaluates to true in $\mathit{cens}(\mathcal{A})$ iff there exists an image M of $\mathit{PerfectRef}(q, \mathcal{T})$ in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ such that $\mathcal{T} \cup \mathcal{P} \cup M$ is consistent. Now, by Proposition 21, $\mathcal{T} \cup \mathcal{P} \cup M$ is consistent iff $\mathit{PerfectRef}(\mathcal{P}, \mathcal{T})$ evaluates to false in M. Consequently, Lemma 10 implies the thesis. ∎

Then, we use $\mathit{BraveRef}(\cdot, \cdot)$ to define a further reformulation function as follows.

Definition 30. [*StateRef*] Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{R}$ CQE specification, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ (with $n \geq 0$) be a sequence of BUCQs, let i be such that $1 \leq i \leq n$, and let $I \subseteq \{1, \ldots, i-1\}$. We define $\mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I)$ as the FO sentence:

$$\left(\bigwedge_{\substack{1 \leq j \leq i-1 \\ \wedge\ j \notin I}} \neg \mathit{BraveRef}\Big(\Big(\bigwedge_{\ell \in I \wedge \ell < j} q_\ell \Big) \wedge q_j, \mathcal{E} \Big) \right) \wedge \mathit{BraveRef}\Big(\Big(\bigwedge_{\ell \in I} q_\ell \Big) \wedge q_i, \mathcal{E} \Big)$$

Example 36. Consider the DL-Lite$_\mathcal{R}$ CQE specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$, the ABox $\mathcal{A}$ and the query sequence $\mathcal{Q} = \langle q_1, q_2, q_3 \rangle$ of Example 34, and let us set $i = 3$ and $I = \{1\}$. We have that:

$$\begin{aligned} \mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I) &= \neg \mathit{BraveRef}(q_1 \wedge q_2, \mathcal{E}) \wedge \mathit{BraveRef}(q_1 \wedge q_3, \mathcal{E}) \\ &= \neg(\mathsf{buy}(\mathsf{john}, \mathsf{m}_a) \wedge \mathsf{BrandX}(\mathsf{m}_a) \wedge \\ &\quad \neg(\exists z, w\, (\mathsf{buy}(z, w) \wedge \mathsf{BrandX}(w) \wedge z = \mathsf{john} \wedge w = \mathsf{m}_a))) \wedge \\ &\quad \exists x\, (\mathsf{buy}(\mathsf{john}, \mathsf{m}_a) \wedge \mathsf{buy}(x, \mathsf{m}_b)). \end{aligned}$$

▽

In Definition 30, the set I represents the indexes of the queries that precede q_i in $\mathcal{Q}$ and that are guessed to be true in the state. The subsequent lemma expresses this intuition in a formal manner.

Lemma 11. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{R}$ CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ be a sequence of BUCQs, let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$, let i be such that $1 \leq i \leq n$, and let $I \subseteq \{1, \ldots, i-1\}$. Then, the FO query StateRef$(\mathcal{E}, \mathcal{Q}, i, I)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ iff the following two conditions hold:*

- *$q_i \in \mathit{EntQ}(\mathcal{S})$;*
- *for each j such that $1 \leq j < i$, $q_j \in \mathit{EntQ}(\mathcal{S})$ iff $j \in I$.*

Proof. ($\Leftarrow$:) First, suppose that $\mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I)$ evaluates to false in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. We have two cases:

(i) the sentence $\mathit{BraveRef}((\bigwedge_{\ell \in I} q_\ell) \wedge q_i, \mathcal{E})$ (i.e. the last conjunct occurring in $\mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I)$) evaluates to false in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. Then, Proposition 24 implies that there exists no optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models (\bigwedge_{\ell \in I} q_\ell) \wedge q_i$, which implies that either $q_i \notin \mathit{EntQ}(\mathcal{S})$ or there exists $j \in I$ such that $q_j \notin \mathit{EntQ}(\mathcal{S})$, thus proving the thesis.

(ii) the sentence $\mathit{BraveRef}((\bigwedge_{\ell \in I} q_\ell) \wedge q_i, \mathcal{E})$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ and, for some j such that $1 \leq j < i$ and $j \notin I$, the conjunct $\neg \mathit{BraveRef}((\bigwedge_{\ell \in I \wedge \ell < j} q_\ell) \wedge q_j, \mathcal{E}))$ evaluates to false in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. Let us assume that j is the least index such that the above property holds. Then, by Proposition 24 there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models (\bigwedge_{\ell \in I \wedge \ell < j} q_\ell) \wedge q_j$, which implies that $q_j \in \mathit{EntQ}(\mathcal{S})$, thus proving the thesis.

($\Rightarrow$:) Now suppose that $\mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. First, we prove by induction that, for each j such that $1 \leq j < i$, $j \in I$ iff $q_j \in \mathit{EntQ}(\mathcal{S})$.

◊ Base case ($j = 1$). We prove that $1 \in I$ iff $q_1 \in \mathit{EntQ}(\mathcal{S})$.

(i) Suppose that $1 \in I$. In this case, since $\mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, also its conjunct $\mathit{BraveRef}((\bigwedge_{\ell \in I} q_\ell) \wedge q_i, \mathcal{E})$ does. Therefore, by Proposition 24 there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models (\bigwedge_{\ell \in I} q_\ell) \wedge q_i$. In particular, the function $\mathit{cens}(\cdot)$ is such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q_1$. Thus, we have that $q_1 \in \mathit{EntQ}(\mathcal{S}_1)$, which by Lemma 8 implies that $q_1 \in \mathit{EntQ}(\mathcal{S})$.

(ii) Suppose that $1 \notin I$. In this case, $\neg \mathit{BraveRef}(q_1, \mathcal{E})$ is one of the conjuncts of $\mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I)$. Then, since the latter evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, also $\neg \mathit{BraveRef}(q_1, \mathcal{E})$ does. Hence $\mathit{BraveRef}(q_1, \mathcal{E})$ evaluates to false in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, which by Proposition 24 implies that there exists no optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models q_1$. Thus, it holds $q_1 \notin \mathit{EntQ}(\mathcal{S})$.

$\Diamond$ Inductive case: suppose that, for each ℓ such that $1 \leq \ell \leq j$ (with $j < i-1$), $\ell \in I$ iff $q_\ell \in \mathit{EntQ}(\mathcal{S})$. We prove that $\ell+1 \in I$ iff $q_{\ell+1} \in \mathit{EntQ}(\mathcal{S})$. There are two cases:

(i) Suppose that $\ell+1 \in I$. In this case, since $\mathit{StateRef}(\mathcal{E}, \mathcal{Q}, \ell+1, I)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, also its conjunct $\mathit{BraveRef}((\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_\ell)} q) \wedge q_{\ell+1}, \mathcal{E})$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. Therefore, Proposition 24 implies that there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models (\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_\ell)} q) \wedge q_{\ell+1}$, which implies that $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S}_{\ell+1})$, therefore $q_{\ell+1} \in \mathit{EntQ}(\mathcal{S}_{\ell+1})$, and hence $q_{\ell+1} \in \mathit{EntQ}(\mathcal{S})$.

(ii) Suppose that $\ell+1 \notin I$. In this case, since $\mathit{StateRef}(\mathcal{E}, \mathcal{Q}, \ell+1, I)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, also its conjunct $\neg \mathit{BraveRef}((\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_\ell)} q) \wedge q_{\ell+1}, \mathcal{E})$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, thus $\mathit{BraveRef}((\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_\ell)} q) \wedge q_{\ell+1}, \mathcal{E})$ evaluates to false in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. Hence, Proposition 24 implies that there is no optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models (\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_\ell)} q) \wedge q_{\ell+1}$, which implies that $q_{\ell+1} \notin \mathit{EntQ}(\mathcal{S})$.

Then, since for each j such that $1 \leq j < i$, $j \in I$ iff $q_j \in \mathit{EntQ}(\mathcal{S})$, and since $\mathit{BraveRef}((\bigwedge_{j \in I} q_j) \wedge q_i, \mathcal{E})$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$, Proposition 24 implies that there exists an optimal GA censor $\mathit{cens}(\cdot)$ for $\mathcal{E}$ such that $\mathcal{T} \cup \mathit{cens}(\mathcal{A}) \models (\bigwedge_{q \in \mathit{EntQ}(\mathcal{S}_{i-1})} q) \wedge q_i$, from which it follows that $\mathit{cens}(\cdot) \in \mathit{stCens}(\mathcal{S}_i)$. Therefore, we have that $q_i \in \mathit{EntQ}(\mathcal{S}_i)$, which by Lemma 8 implies that $q_i \in \mathit{EntQ}(\mathcal{S})$, thus proving the thesis. ∎

Since the set I of indexes in Definition 30 is not known a priori, the ultimate rewriting covers all the possible subsets I of $\{1, \ldots, i-1\}$. This can be done independently on the input ABox, allowing us to reduce the dynCQE query entailment to evaluating an FO query in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. By exploiting the $\mathit{AtomRewr}(\cdot, \cdot)$ function introduced in Section 3.3.1, we can then extend this property in order to produce a "genuine" FO rewritability result, as stated by the following theorem.

Theorem 15. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{R}$ CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n \rangle$ be a sequence of BUCQs, and let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \rangle$. For every i such that $1 \leq i \leq n$, we have that $q_i \in \mathit{EntQ}(\mathcal{S})$ iff the following FO sentence evaluates to true in $\mathcal{A}$.*[2]

$$\mathit{AtomRewr}\Big(\bigvee_{I \in \wp(\{1,\ldots,i-1\})} \mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I), \mathcal{T}\Big)$$

Proof. Consider the subformula $\bigvee_{I \in \wp(\{1,\ldots,i-1\})} \mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I)$, denoted by q_r. Clearly, q_r evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ iff there exists a subset I of $\{1, \ldots, i-1\}$

[2] We use the notation $\wp(S)$ for indicating the power set of a set S.

such that $\mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I)$ evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$. Then, Lemma 11 implies that q_r evaluates to true in $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A})$ iff there exists $q_i \in \mathit{EntQ}(\mathcal{S})$. Finally, the thesis immediately follows from Lemma 5. ∎

Corollary 5. *dynCQE-Ent for DL-Lite$_\mathcal{R}$ CQE specifications and sequences of BUCQs is in AC^0 in data complexity.*

Example 37. Let $\mathcal{E}$, $\mathcal{A}$ and $\mathcal{Q} = \langle q_1, q_2, q_3\rangle$ be as in Example 34. According to Theorem 15, the query $q_3 = \exists x\, \mathsf{buy}(x, \mathsf{m}_b)$ belongs to $\mathit{EntQ}(\mathcal{S}_3)$ iff the FO sentence below evaluates to true in $\mathcal{A}$ (f_I denotes the sub-formula considering the guess I of the indexes of the queries that precede the query q_3):

$$\mathit{AtomRewr}(\textstyle\bigvee_{I\in\wp(\{1,2\})} \mathit{StateRef}(\mathcal{E}, \mathcal{Q}, i, I), \mathcal{T}) =$$

$$\begin{array}{ll}
f_\emptyset & \neg \mathit{BraveRef}(q_1, \mathcal{E}) \wedge \neg \mathit{BraveRef}(q_2, \mathcal{E}) \wedge \mathit{BraveRef}(q_3, \mathcal{E}) \vee \\
f_{\{1\}} & \neg \mathit{BraveRef}(q_1 \wedge q_2, \mathcal{E}) \wedge \mathit{BraveRef}(q_1 \wedge q_3, \mathcal{E}) \vee \\
f_{\{2\}} & \neg \mathit{BraveRef}(q_1, \mathcal{E}) \wedge \mathit{BraveRef}(q_2 \wedge q_3, \mathcal{E}) \vee \\
f_{\{1,2\}} & \mathit{BraveRef}(q_1 \wedge q_2 \wedge q_3, \mathcal{E}) = \\
f_\emptyset & \neg \mathsf{buy}(\mathsf{john}, \mathsf{m}_a) \wedge \neg \mathsf{BrandX}(\mathsf{m}_a) \wedge \exists x\, \mathsf{buy}(x, \mathsf{m}_b) \vee \\
f_{\{1\}} & \neg(\mathsf{buy}(\mathsf{john}, \mathsf{m}_a) \wedge \mathsf{BrandX}(\mathsf{m}_a) \wedge \neg(\exists z, w\, (\mathsf{buy}(z, w) \wedge \mathsf{BrandX}(w) \wedge \\
 & \qquad z = \mathsf{john} \wedge w = \mathsf{m}_a))) \wedge \exists x\, (\mathsf{buy}(\mathsf{john}, \mathsf{m}_a) \wedge \mathsf{buy}(x, \mathsf{m}_b)) \vee \\
f_{\{2\}} & \neg(\mathsf{buy}(\mathsf{john}, \mathsf{m}_a)) \wedge (\exists x\, (\mathsf{BrandX}(\mathsf{m}_a) \wedge \mathsf{buy}(x, \mathsf{m}_b))) \vee \\
f_{\{1,2\}} & \exists x\, (\mathsf{buy}(\mathsf{john}, \mathsf{m}_a) \wedge \mathsf{BrandX}(\mathsf{m}_a) \wedge \mathsf{buy}(x, \mathsf{m}_b)) \wedge \\
 & \qquad \neg(\exists z, w\, (\mathsf{buy}(z, w) \wedge \mathsf{BrandX}(w) \wedge z = \mathsf{john} \wedge w = \mathsf{m}_a))
\end{array}$$

which, indeed, evaluates to true in $\mathcal{A}$ thanks to $f_{\{1\}}$. ▽

7.3 Towards practical techniques and approximations

We now provide a simplification of the query rewriting presented in Theorem 15. In particular, in a real maximally collaborative CQE system, the answers to the queries already executed (i.e. the queries belonging to the state) can obviously be stored and re-used when the next query is submitted. This allows for greatly simplifying the structure of the FO reformulation of the query defined in Theorem 15, as shown in the following.

Theorem 16. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{P}\rangle$ be a DL-Lite$_\mathcal{R}$ CQE specification, let $\mathcal{A}$ be an ABox, let $\mathcal{Q} = \langle q_1, \ldots, q_n\rangle$ be a sequence of BUCQs, let $\mathcal{S} = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q}\rangle$, let q_{n+1} be a BUCQ, and let $\mathcal{S}' = \langle \mathcal{E}, \mathcal{A}, \mathcal{Q} \circ \langle q_{n+1}\rangle\rangle$. Then, q_{n+1} is entailed by $\mathcal{S}'$ iff the following FO sentence evaluates to true in $\mathcal{A}$:*

$$\mathit{AtomRewr}\Big(\mathit{BraveRef}\Big(\Big(\bigwedge_{q\in \mathit{EntQ}(\mathcal{S})} q\Big) \wedge q_{n+1}, \mathcal{E}\Big), \mathcal{T}\Big)$$

Proof. In the following, we indicate with $\mathcal{Q}'$ the sequence of BUCQs $\mathcal{Q} \circ \langle q_{n+1}\rangle$.

($\Rightarrow$:) Suppose $\mathcal{S}' \models q_{n+1}$, i.e. $\mathit{EntQ}(\mathcal{S}') = \mathit{EntQ}(\mathcal{S}) \cup \{q_{n+1}\}$. By Theorem 15, the sentence $\psi = \mathit{AtomRewr}(\mathit{StateRef}(\mathcal{E}, \mathcal{Q}', n+1, I), \mathcal{T})$ evaluates to true in $\mathcal{A}$, where $I = \{i \mid q_i \in \mathit{EntQ}(\mathcal{S}')\}$. Consequently, $\mathit{AtomRewr}(\mathit{BraveRef}((\bigwedge_{q_i\in \mathit{EntQ}(\mathcal{S})} q_i) \wedge$

$q_{n+1}, \mathcal{E}), \mathcal{T})$ is equal to the last conjunct of ψ, and therefore evaluates to true in $\mathcal{A}$ as well.

($\Leftarrow$:) Suppose now $\mathcal{S}' \not\models q_{n+1}$. From Theorem 15, we have that the sentence $\mathit{AtomRewr}(\mathit{StateRef}(\mathcal{E}, \mathcal{Q}', n+1, I), \mathcal{T})$ evaluates to false in $\mathcal{A}$, where $I = \{i \mid q_i \in \mathit{EntQ}(\mathcal{S})\}$. Since $\mathit{EntQ}(\mathcal{S})$ is the set of BUCQ from $\langle q_1, \ldots, q_n\rangle$ entailed by $\mathcal{S}$, all the conjuncts of $\mathit{AtomRewr}(\mathit{StateRef}(\mathcal{E}, \mathcal{Q}', n+1, I), \mathcal{T})$ except the last one evaluate to true in $\mathcal{A}$. This means that its last conjunct evaluates to false in $\mathcal{A}$. Such a conjunct is equal to the sentence $\mathit{AtomRewr}(\mathit{BraveRef}((\bigwedge_{q_i \in \mathit{EntQ}(\mathcal{S})} q_i) \wedge q_{n+1}, \mathcal{E}), \mathcal{T})$, which proves the thesis. ∎

Example 38. Let $\mathcal{E}$, $\mathcal{A}$ and the queries q_1, q_2, and q_3 be as in Example 34, and consider the sequence of queries $\mathcal{Q} = \langle q_1, q_2\rangle$. From Example 34, we know that only $q_1 = \mathsf{buy}(\mathsf{john}, \mathsf{m}_a)$ belongs to $\mathit{EntQ}(\langle \mathcal{E}, \mathcal{A}, \mathcal{Q}\rangle)$. Hence, according to Theorem 16, the query $q_3 = \exists x\, \mathsf{buy}(x, \mathsf{m}_b)$ is entailed by the state $\langle \mathcal{E}, \mathcal{Q} \circ \{q_3\}\rangle$ iff the FO sentence $\exists x\, (\mathsf{buy}(\mathsf{john}, \mathsf{m}_a) \wedge \mathsf{buy}(x, \mathsf{m}_b))$ evaluates to true in $\mathcal{A}$. ▽

An issue that the query rewriting technique of Theorem 16 does not solve is the scalability with respect to the number of submitted queries, which might become too large to make the FO query produced by the rewriting executable in practice. On the other hand, Theorem 14 shows that it is not always possible to intensionally simulate the dynCQE semantics by using a stateless CQE specification, i.e. through an ABox-independent transformation of the intensional part of a CQE specification.

To overcome the above issue, a possible approach is to choose a censor $\mathit{cens}(\cdot)$ of the current state $\mathcal{S}$ of the CQE specification, materialize the output of $\mathit{cens}(\mathcal{A})$, and then check if next queries are entailed by the ontology $\mathcal{T} \cup \mathit{cens}(\mathcal{A})$. Obviously, if all the censors of the state $\mathcal{S}$ yield the same outcome for the input ABox $\mathcal{A}$, then query entailment over $\mathcal{T} \cup \mathit{cens}(\mathcal{A})$ conforms to the dynCQE semantics independently on the chosen censor $\mathit{cens}(\cdot)$.

On the other hand, if the current state $\mathcal{S}$ has at least another censor $\mathit{cens}'(\cdot)$ such that $\mathit{cens}(\mathcal{A}) \neq \mathit{cens}'(\mathcal{A})$, then the answers to next queries will only approximate query entailment through dynCQE, i.e. in the corresponding state. More precisely: as long as the materialized system processes only queries entailed by $\mathcal{T} \cup \mathit{cens}(\mathcal{A})$ (i.e. it always answers *true*), it returns exactly the same answers provided by the dynCQE semantics. The first time it processes a query q not entailed by $\mathcal{T} \cup \mathit{cens}(\mathcal{A})$ (i.e. it answers $false$), its behavior might differ from the dynamic approach, where q might be either entailed or not entailed (depending on how the censors of the states evolve). After the first negative answer, the system using $\mathit{cens}(\mathcal{A})$ might answer *true* (resp., $false$) to a subsequent query q even if the corresponding state does not entail (resp., entails) q.

We now describe how to materialize the ABox $\mathit{cens}(\mathcal{A})$ for some censor $\mathit{cens}(\cdot)$ of a given state $\mathcal{S} = \langle \mathcal{T}, \mathcal{Q}\rangle$. First note that, by Theorem 16 and since by definition every query $q \in \mathit{EntQ}(\mathcal{S})$ is entailed by $\mathcal{S}$, then the following FO sentence evaluates to true in $\mathcal{A}$:

$$\mathit{AtomRewr}\Big(\mathit{BraveRef}\Big(\bigwedge_{q \in \mathit{EntQ}(\mathcal{S})} q, \mathcal{E}\Big), \mathcal{T}\Big).$$

Note also that the above formula can be straightforwardly turned into a disjunction ϕ of BCQ$^{\neq}$s. Then, one can proceed as follows:

1. consider the set of BCQ$^{\neq}$s occurring in ϕ, remove from such queries all the existential quantifiers and execute them one at a time;

2. as soon as the answer set of one of such queries $\mathcal{A}$ is not empty, we can construct (through the corresponding binding of the free variables of the query) an image of this query in $\mathcal{A}$. Let $\mathcal{A}'$ be such a subset of $\mathcal{A}$;

3. $\mathcal{P} \cup \mathcal{A}'$ is consistent, so there exists at least one censor of $\mathcal{S}$ that contains $\mathcal{A}'$. The materialization of $cens(\mathcal{A})$ for one such censor $cens(\cdot)$ can be computed by first setting $\mathcal{C} = \mathcal{A}'$, and then, as long as it is possible, by iteratively adding to $\mathcal{C}$ ground atoms α from $cl^{\mathcal{T}}_{\mathbf{GA}}(\mathcal{A}) \setminus \mathcal{A}'$ such that $\mathcal{T} \cup \mathcal{P} \cup \mathcal{C} \cup \{\alpha\}$ is consistent.

Chapter 8

Consistent query answering

This chapter deals with a distinct topic from CQE, namely *consistent query answering* (*CQA*). As anticipated in the introduction, in contrast with our treatment until this point, we now adopt the so-called *closed-world assumption* (*CWA*), which involves always interpreting as false each fact not belonging to the original database. In practice, this affects the way we employ the logical rules, i.e. not for inferring new knowledge but for expressing constraints on the data.

The first part of this chapter defines syntactically all the types of rules (dependencies) adopted, as well as the decision problems under consideration. The subsequent section delves into a detailed analysis of these problems with respect to the various classes of dependencies introduced.

8.1 Dependencies

We recall that a database instance (or simply database) is defined as any set of ground atoms. Hereinafter, we assume w.l.o.g. that the predicate signature of every database always contains the special predicate $\bot$ of arity 0, and that no database contains the fact $\bot$. We also assume that the evaluation of such a fact $\bot$ is false under any FO interpretation.

We now define the notion of dependency that we use throughout this chapter for studying CQA problems.[1]

Definition 31. [Dependency] A *disjunctive embedded dependency with inequalities* (DED$^{\neq}$), or simply *dependency*, is an FO sentence of the form:

$$\forall \mathbf{x} \, (\gamma \rightarrow q), \tag{8.1}$$

where $\mathbf{x}$ is a sequence of variables such that $\exists \mathbf{x} \, (\gamma)$ is a BCQ$^{\neq}$, and q is a UCQ$^{\neq}$ whose free variables occur in $\mathbf{x}$, and such that $q(\mathbf{t})$ is a BUCQ$^{\neq}$ for any tuple $\mathbf{t}$ of constants of the same arity as $\mathbf{x}$.

Given a dependency τ of the above form (8.1), we indicate with $\mathit{body}(\tau)$ the CQ$^{\neq}$ γ and with $\mathit{head}(\tau)$ the UCQ$^{\neq}$ $q = \bigvee_{i=1}^{m} q_i$. Moreover, we say that τ is

[1] Although slightly different, Definition 31 is actually equivalent to the notion of disjunctive embedded dependency with inequalities *and equalities* presented in [62, 63].

non-disjunctive if $m = 1$, i.e. $head(\tau)$ is a $\text{CQ}^{\neq}$, and that τ is *single-head* if it is non-disjunctive and $|predAt(head(\tau))| = 1$.

We remark that $\text{DED}^{\neq}$s is a sublanguage of domain-independent relational calculus formulas. Given a database $\mathcal{D}$ and a dependency τ, we say that τ is *satisfied by* $\mathcal{D}$ if it evaluates to true in $\mathcal{D}$. Given a database $\mathcal{D}$ and a set of dependencies Σ, we say that $\mathcal{D}$ is *satisfies* Σ if all the dependencies of Σ are satisfied by $\mathcal{D}$, i.e. if $\bigwedge_{\tau \in \Sigma} \tau$ evaluates to true in $\mathcal{D}$. From this definition we immediately have that:

Proposition 25. *Deciding if a database satisfies a set of dependencies is in* AC^0 *in data complexity.*

The following property, which is immediate to verify, is implicitly leveraged throughout this chapter.

Proposition 26. *Let* τ *be a dependency, and let* $\mathcal{D}$ *be a database. Then,* τ *is satisfied by* $\mathcal{D}$ *iff, for every instantiation* σ *of* $body(\tau)$ *in* $\mathcal{D}$*, there exists an image of* $head(\sigma(\tau))$ *in* $\mathcal{D}$*.*

We point out that the images of a $\text{BCQ}^{\neq}$ with k predicate atoms in a set of facts $\mathcal{D}$ are at most n^k (where n is the cardinality of $\mathcal{D}$) and can be computed in polynomial time with respect to the size of the database.

8.1.1 Subclasses of dependencies

We say that a dependency of the form (8.1) is *linear* if $|predAt(\gamma)| = 1$. Also, we call *tuple-generating dependency* (*TGD*) a non-disjunctive dependency having no occurrences of inequality atoms. Moreover, if $|CQ(q)| = 1$ and $vars(q) \subseteq vars(\gamma)$, then we say that the dependency is *full non-disjunctive*, or simply *full*. If $head(\tau) = \bot$, then τ is a *denial*. Note that, in this chapter, denials may have inequalities, and that every denial dependency is full.

Hereinafter, we assume w.l.o.g. that each full dependency τ is such that $head(\tau)$ consists of a single predicate atom. Observe indeed that:

- it is always possible to equivalently represent a full dependency through a set of full dependencies with a single atom in their head (such a set contains one dependency for each atom in the head of the initial dependency);
- each τ of the form $\forall \mathbf{x}\,(\varphi(\mathbf{x}) \rightarrow x \neq t)$[2] (with $x \in \mathbf{x}$) can be rewritten as $\forall \mathbf{x}\,(\sigma(\varphi(\mathbf{x})) \rightarrow \bot)$, where σ is the substitution $\{x \mapsto t\}$.

Given a set of dependencies Σ, we call *dependency graph* of Σ the directed graph $\mathcal{G}(\Sigma)$ whose vertices are the dependencies of Σ and such that there is one edge from the vertex τ_1 to τ_2 iff the head of τ_1 contains an atom whose predicate appears in a predicate atom of the body of τ_2. We say that Σ is *acyclic* if there is no cyclic path in $\mathcal{G}(\Sigma)$. Given a set of acyclic dependencies Σ, we call *topological order of* Σ any topological order of $\mathcal{G}(\Sigma)$, i.e. a sequence $\langle \tau_1, \ldots, \tau_h \rangle$ of the dependencies of Σ such that, if $i \geq j$, then the vertex τ_i is not reachable from the vertex τ_j in $\mathcal{G}(\Sigma)$.

[2] We assume w.l.o.g. that no inequality of the form $t \neq t$ or $c \neq d$, where c and d are distinct constants, appears in τ.

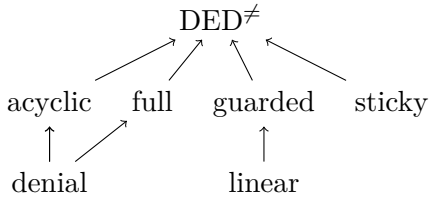

Figure 8.1. Illustration of hierarchy for sets of dependencies.

We then recall the classes of guarded and sticky dependencies [40, 12, 41]. A dependency τ is *guarded* if there exists an atom α of $body(\tau)$ such that $vars(\alpha) = vars(body(\tau))$. Obviously, if a set of dependencies is linear, it is also guarded. As usual, in order to introduce the stickiness property, we make use of the following auxiliary inductive definition. Given a set of dependencies Σ, for each dependency $\tau \in \Sigma$ we say that a variable x occurring in $body(\tau)$[3] is *marked* if:

(i) there exists an atom of $head(\tau)$ in which x does not occur, or

(ii) there exist a dependency τ' and a predicate p such that $head(\tau)$ and $body(\tau')$ contain, respectively, $p(\mathbf{t})$ and $p(\mathbf{t}')$ and, for every position i of $\mathbf{t}$ in which x occurs, the i-th term of $\mathbf{t}'$ is a marked variable.

Then, we say that Σ is *sticky* if, for every dependency $\tau \in \Sigma$, every marked variable occurs in $body(\tau)$ at most once.

8.1.2 The chase

We now recall the notion of *chase* for TGDs [68]. Formally, given a database $\mathcal{D}$ and a set of TGDs and denials Σ, we write $Chase(\mathcal{D}, \Sigma)$ for indicating the output of the following procedure. First, initialize a set of predicate atoms $\mathcal{C}$ with the content of $\mathcal{D}$. Then, for each dependency τ in Σ, if there exists an instantiation σ of $body(\tau)$ in $\mathcal{C}$ such that $\sigma(head(\tau))$ evaluates to false in $\mathcal{C}$, then add to $\mathcal{C}$ the atoms of $\sigma(head(\tau))$, where distinct variables are replaced with distinct fresh *labelled nulls* (i.e. special constants that do not already occur in $\mathcal{C}$). It is well-known that such a procedure may not terminate. In particular, termination is not guaranteed for cyclic and non-full sets of dependencies.

In the rest of the thesis, sometimes we apply the chase to a set Σ of full dependencies, that is not a set of strict TGDs because of the presence of either inequalities in the bodies of dependencies and/or the special predicate $\bot$ in the head of dependencies. In these cases, we assume that the chase considers both the inequality and $\bot$ as every other predicate: in this way, Σ is treated as a set of ordinary TGDs. We will see that such a (generally incorrect) treatment of inequalities and $\bot$ is correct in the contexts in which we use the chase.

8.2 Repairs and decision problems

We are now ready to define the notion of repair, the different entailment semantics, and the decision problems that we study. The definition of repair is equivalent to

[3] We assume w.l.o.g. that distinct dependencies use distinct variables.

the one given in [51] and to the one of "subset-repair" given in [2, 114].

Definition 32. [Repair] Given a database $\mathcal{D}$ and a set of dependencies Σ, a *repair* of $\langle \Sigma, \mathcal{D} \rangle$ is a maximal (w.r.t. set inclusion) subset of $\mathcal{D}$ that satisfies Σ.

Let us call $\mathsf{rep}_\Sigma(\mathcal{D})$ the set of all the possible repairs of $\langle \Sigma, \mathcal{D} \rangle$. It is immediate to see that a repair always exists for every $\langle \Sigma, \mathcal{D} \rangle$ (since the empty database satisfies every dependency). Moreover, we denote as $\mathsf{intRep}_\Sigma(\mathcal{D})$ the set $\bigcap_{\mathcal{D}' \in \mathsf{rep}_\Sigma(\mathcal{D})} \mathcal{D}'$.

The main decision problems that we define in this chapter are related to two distinct notions of entailment of formulas with respect to the repairs of a database.

Definition 33. [AR and IAR-entailment] Given as input a database $\mathcal{D}$, a set of dependencies Σ, and a BUCQ$^{\neq}$ q:

(*i*) AR-Ent is the problem of deciding whether q evaluates to true in every $\mathcal{D}' \in \mathsf{rep}_\Sigma(\mathcal{D})$. In case this holds, we write $\langle \Sigma, \mathcal{D} \rangle \models^c_{\text{AR}} q$;

(*ii*) IAR-Ent is the problem of deciding whether q evaluates to true in $\mathsf{intRep}_\Sigma(\mathcal{D})$. In case this holds, we write $\langle \Sigma, \mathcal{D} \rangle \models^c_{\text{IAR}} q$.

The names of these entailment semantics come from the field of Description Logics. In fact, AR and IAR stand for "ABox repair" and "intersection of ABox repairs", respectively [80]. Moreover, IAR-entailment is a well-known sound approximation of AR-entailment. This is illustrated in the subsequent example.

Example 39. Consider the following database and set of dependencies:

$$\mathcal{D} = \{\ T(a), T(b), P(c,a), P(c,b), P(d,c)\ \},$$
$$\Sigma = \{\ \forall x,y,z\,(P(x,y) \wedge P(x,z) \wedge y \neq z \rightarrow \bot), \forall x\,(T(x) \rightarrow \exists y\, P(y,x))\ \}.$$

The first dependency of Σ is not satisfied by $\mathcal{D}$, and there are two minimal ways of solving the inconsistency between $\mathcal{D}$ and Σ. The first one consists in deleting the fact $P(c,a)$ and, due to the second dependency, also the fact $T(a)$. Analogously, the second way involves deleting both $P(c,b)$ and $T(b)$. Thus, we have that:

$$\mathsf{rep}_\Sigma(\mathcal{D}) = \{\ \{P(c,a), P(d,c), T(a)\}, \{P(c,b), P(d,c), T(b)\}\ \}$$

and $\mathsf{intRep}_\Sigma(\mathcal{D}) = \{P(d,c)\}$. Let us now consider the query $q = \exists x\, P(c,x)$. We have that q is entailed by all the repairs, but not by their intersection, i.e. we have $\langle \Sigma, \mathcal{D} \rangle \models^c_{\text{AR}} q$ and $\langle \Sigma, \mathcal{D} \rangle \not\models^c_{\text{IAR}} q$. ▽

Our analysis focuses on entailment of BUCQ$^{\neq}$s under both semantics. The following property, whose proof is immediate, shows an important correspondence between the AR and the IAR semantics.

Proposition 27. *In case the query is a ground atom, AR-Ent coincides with IAR-Ent.*

When the query consists in a ground atom, we hence study a unique decision problem, that we call Instance-Checking.

A further correspondence between the two entailment semantics is provided in the following statement, which is a direct consequence of [114, Corollary 4.3].

Proposition 28. *In case of linear dependencies, AR-Ent coincides with IAR-Ent.*

We now define another pivotal decision problem studied in the context of CQA.

Definition 34. [Repair checking] Given as input a set of dependencies Σ and two databases $\mathcal{D}$ and $\mathcal{D}'$, Rep-Checking is the problem of deciding whether $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$.

From now on, in all the decision problems we study, we assume w.l.o.g. that all the predicates occurring in $\mathcal{D}$ also occur in Σ.

8.3 Recoverability

In what follows, we present some auxiliary outcomes that are helpful in proving certain complexity results presented further on. In particular, we introduce the *recoverability* property, that is related to the notion of satisfaction of a set of dependencies by a database.

Definition 35. [Recoverability] Given a set of dependencies Σ, we say that a set of facts $\mathcal{D}'$ is *recoverable from* $\langle \Sigma, \mathcal{D} \rangle$ if there exists a subset $\mathcal{D}''$ of $\mathcal{D}$ such that $\mathcal{D}' \subseteq \mathcal{D}''$ and $\mathcal{D}''$ satisfies Σ.

Intuitively, a set of facts $\mathcal{D}'$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$ if it is a subset of $\mathcal{D}$ that can be "repaired" by adding facts belonging to the original database $\mathcal{D}$. Obviously, if $\mathcal{D}'$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$ then it is contained in some repair of $\langle \Sigma, \mathcal{D} \rangle$. For some classes of dependencies, we then study the associated decision problem of checking whether a set of facts $\mathcal{D}'$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$, which we indicate as Recoverability.

We now turn our attention to full dependencies. We recall that the chase of a database w.r.t. full dependencies can be computed in polynomial time. Consequently, the following property holds.

Lemma 12. *Recoverability is in PTime in data complexity in the case of full dependencies.*

Proof. It is straightforward to verify that a database $\mathcal{D}'$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$ iff $\mathit{Chase}(\mathcal{D}', \Sigma) \subseteq \mathcal{D}$. Then, the thesis follows from the fact that, for full dependencies, $\mathit{Chase}(\mathcal{D}', \Sigma)$ is finite and can be computed in polynomial time. ∎

Consider now the following crucial property.

Definition 36. [CQ-FO-rewritability] We call a set of dependencies Σ *CQ-FO-rewritable* if, for every CQ $q(\mathbf{x})$ there exists an FO query $\phi_q(\mathbf{x})$ such that, for every database $\mathcal{D}$ and for every tuple of constants $\mathbf{t}$ of the same arity as $\mathbf{x}$, $\Sigma \cup \mathcal{D} \models q(\mathbf{t})$ iff $\phi_q(\mathbf{t})$ evaluates to true in $\mathcal{D}$.

It is easy to see that CQ-FO-rewritability is enjoyed by any set of full dependencies if such a set is also acyclic, or linear, or sticky, as shown below.

Proposition 29. *Every set of acyclic+full dependencies, every set of full+linear dependencies, and every set of full+sticky dependencies is CQ-FO-rewritable.*

Proof. To prove the proposition, we introduce some auxiliary definitions. Given a single-head full dependency τ, we define τ_t as the TGD obtained from τ deleting all the inequality atoms in its body. Moreover, given a set of full dependencies Σ, we denote by $TGD(\Sigma)$ the set of full TGDs $\{\tau_t \mid \tau \in \Sigma\}$.

Now, the key observation is that it is very easy to modify existing rewriting techniques for conjunctive queries over TGDs to deal with single-head full dependencies without inequalities. E.g. the algorithm shown in [77] can very easily be adapted to such single-head full dependencies with inequality atoms in rule bodies. Indeed, since the dependencies are full, no variable is deleted by the rewriting steps, which implies that the rewriting of every CQ over Σ can be obtained essentially in the same way as in the case of the corresponding $TGD(\Sigma)$: the additional inequality atoms can just be collected along the rewriting steps, and returned in the final rewritten query (which is thus a UCQ with inequalities rather than a simple UCQ). This immediately implies the following property.

Lemma 13. *Let Σ be a set of full dependencies. If $TGD(\Sigma)$ is CQ-FO-rewritable, then Σ is CQ-FO-rewritable.*

Now, since acyclic TGDs, linear TGDs, and sticky TGDs have the CQ-FO-rewritability property (see e.g. [40, 41]), the above lemma immediately implies that the CQ-FO-rewritability property holds for every set of dependencies that is either acyclic+full, or full+linear, or full+sticky. ∎

We now prove that, for the above classes of dependencies, checking recoverability is FO rewritable, i.e. it is always possible to build an FO formula (depending only on Σ) whose evaluation decides if $\mathcal{D}'$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$. To this aim, in order to represent the subset $\mathcal{D}'$ of $\mathcal{D}$, we make use of an auxiliary alphabet that contains a predicate symbol $\overset{*}{p}$ for every $p \in pred(\Sigma)$, such that the predicates $\overset{*}{p}$ and p have the same arity. Given a formula ϕ, we denote by $aux(\phi)$ the formula obtained from ϕ replacing every predicate atom $p(\mathbf{t})$ with $\overset{*}{p}(\mathbf{t})$, i.e. the atom obtained from $p(\mathbf{t})$ by replacing its predicate with the corresponding predicate in the auxiliary alphabet. Similarly, given a set of predicate atoms $\mathcal{A}$, we denote by $aux(\mathcal{A})$ the set $\{aux(\alpha) \mid \alpha \in \mathcal{A}\}$.

Now, let Σ be a CQ-FO-rewritable set of dependencies. We define the following formula:

$$\Phi^{rec}(\Sigma) = \bigwedge_{p \in pred(\Sigma)} \forall \mathbf{x} \, (q_p^{\Sigma}(\mathbf{x}) \rightarrow p(\mathbf{x}))$$

where $\mathbf{x} = x_1, \ldots, x_k$ if k is the arity of p, and $q_p^{\Sigma}(\mathbf{x})$ is the FO-rewriting of the atomic query $p(\mathbf{x})$ (with free variables $\mathbf{x}$) with respect to Σ, expressed over the auxiliary predicate symbols. Intuitively, by evaluating this sentence over a database $\mathcal{D} \cup aux(\mathcal{D}')$ one is checking whether $\mathcal{D}$ contains all the facts that are logical consequences of $\Sigma \cup \mathcal{D}'$.

For checking recoverability of a database, we have to slightly modify the above formula as follows. Given a set of predicate atoms $\mathcal{A}$ over the auxiliary predicates, we define $\Phi^{rec}(\mathcal{A}, \Sigma)$ as the formula (with free variables $vars(\mathcal{A})$) obtained from $\Phi^{rec}(\Sigma)$ by replacing every occurrence of every auxiliary atom $\overset{*}{p}(\mathbf{t})$ with the formula $(\overset{*}{p}(\mathbf{t}) \vee \bigvee_{\overset{*}{p}(\mathbf{t}') \in \mathcal{A}} \mathbf{t} = \mathbf{t}')$, where $\mathbf{t} = \mathbf{t}'$ is the conjunction of equalities between the terms of $\mathbf{t}$ and $\mathbf{t}'$. For example, if $\Phi^{rec}(\Sigma)$ is the sentence $\forall x_1, x_2 \, (\overset{*}{s}(x_1, x_2) \vee \overset{*}{r}(x_2, x_1) \rightarrow$

$r(x_1, x_2)$), then $\Phi^{rec}(\{\overset{*}{r}(y_1, y_2)\}, \Sigma)$ is the sentence $\forall x_1, x_2 \, (\overset{*}{s}(x_1, x_2) \vee (\overset{*}{r}(x_2, x_1) \vee (y_1 = x_2 \wedge y_2 = x_1)) \rightarrow r(x_1, x_2))$. For catching the intuition behind such a formula, suppose all the atoms of $\mathcal{A}$ are ground. Then, the equalities introduced in $\Phi^{rec}(\Sigma)$ are used for simulating the presence of the facts from $\mathcal{A}$ in the database the formula is evaluated over. In this way, evaluating $\Phi^{rec}(\mathcal{A}, \Sigma)$ over $\mathcal{D} \cup \mathit{aux}(D')$ is equivalent to evaluating $\Phi^{rec}(\Sigma)$ over $\mathcal{D} \cup \mathit{aux}(D') \cup \mathcal{A}$.

It is now straightforward to verify that the following property holds.

Lemma 14. *Let Σ be a CQ-FO-rewritable set of full dependencies, and let $\mathcal{D}$ and $\mathcal{D}'$ be two databases. Then, $\Phi^{rec}(\Sigma)$ evaluates to true in $\mathcal{D} \cup \mathit{aux}(\mathcal{D}')$ iff $\mathit{Chase}(\mathcal{D}', \Sigma) \subseteq \mathcal{D}$. Moreover, for every set of predicate atoms $\mathcal{A}$ and for every instantiation σ of $\bigwedge_{\alpha \in \mathcal{A}} \alpha$ in $\mathcal{D}$, $\sigma(\Phi^{rec}(\mathcal{A}, \Sigma))$ evaluates to true in $\mathcal{D} \cup \mathit{aux}(\mathcal{D}')$ iff $\mathit{Chase}(\mathcal{D}' \cup \sigma(\mathcal{A}), \Sigma) \subseteq \mathcal{D}$.*

Corollary 6. *Recoverability is in AC^0 in data complexity in the case of CQ-FO-rewritable sets of full dependencies.*

An example of application of the rewriting function $\Phi^{rec}(\cdot)$ is illustrated below.

Example 40. In a university, every lecturer (predicate L) of some course that is attended (predicate A) by a student is also registered as a teacher (predicate T) for that student. Moreover, nobody can be registered as a teacher of herself. We also know that Bob is attending the course c_1 that is taught by Tom, while Ann is attending the course c_2 that is taught by herself. All this can be modeled through the following database and set of acyclic+full dependencies:

$$\mathcal{D} = \{A(\mathsf{bob}, \mathsf{c}_1), L(\mathsf{tom}, \mathsf{c}_1), T(\mathsf{tom}, \mathsf{bob}), \; A(\mathsf{ann}, \mathsf{c}_2), L(\mathsf{ann}, \mathsf{c}_2), T(\mathsf{ann}, \mathsf{ann})\},$$
$$\Sigma = \{\forall l, c, s \, (L(l, c) \wedge A(s, c) \rightarrow T(l, s)), \; \forall t \, (T(t, t) \rightarrow \bot)\}.$$

Then, the sentence $\Phi^{rec}(\Sigma)$ is equal to:

$$\forall l, c, s \, (\overset{*}{L}(l, c) \wedge \overset{*}{A}(s, c) \rightarrow T(l, s)) \wedge \forall t \, (\overset{*}{T}(t, t) \vee \exists c \, (\overset{*}{L}(t, c) \wedge \overset{*}{A}(t, c)) \rightarrow \bot).$$

Now, let us consider two subsets of $\mathcal{D}$, namely $\mathcal{D}' = \{A(\mathsf{bob}, \mathsf{c}_1), L(\mathsf{tom}, \mathsf{c}_1)\}$ and $\mathcal{D}'' = \{A(\mathsf{ann}, \mathsf{c}_2), L(\mathsf{ann}, \mathsf{c}_2)\}$. One can verify that $\Phi^{rec}(\Sigma)$ evaluates to true in $\mathcal{D} \cup \mathit{aux}(\mathcal{D}')$, but it evaluates to false in $\mathcal{D} \cup \mathit{aux}(\mathcal{D}'')$. Indeed, we have that $\mathcal{D}'$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$, while $\mathcal{D}''$ is not. ▽

For the rest of this section, we focus on linear dependencies. We first present Algorithm 4, which extends the one presented in [114, Theorem 4.4]. Such a procedure, given a set of linear dependencies Σ and a database $\mathcal{D}$, computes the unique repair of $\langle \Sigma, \mathcal{D} \rangle$.

It is immediate to verify that the algorithm is correct and runs in polynomial time. It is also evident that a subset $\mathcal{D}'$ of $\mathcal{D}$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$ iff $\mathcal{D}'$ is a subset of the unique repair of $\langle \Sigma, \mathcal{D} \rangle$. Consequently, we derive the following upper bound.

Lemma 15. *Recoverability is in PTime in data complexity in the case of linear dependencies.*

Algorithm 4 *Compute-Repair-Linear*

input: A database $\mathcal{D}$, a set of dependencies Σ;
output: A database $\mathcal{D}'$;
1: $\mathcal{D}' \leftarrow \mathcal{D}$;
2: **repeat**
3: **for all** dependencies $\tau \in \Sigma$ **do**
4: **for all** instantiations σ of $body(\tau)$ in $\mathcal{D}'$ such that $\sigma(head(\tau))$ has no image in $\mathcal{D}'$ **do**
5: $\mathcal{D}' \leftarrow \mathcal{D}' \setminus predAt(\sigma(body(\tau)))$;
6: **until** a fixpoint for $\mathcal{D}'$ is reached;
7: **return** $\mathcal{D}'$;

Finally, we analyze the case of acyclic+linear dependencies and show that the problem of checking recoverability is in AC^0 for such a class of dependencies. In a similar way as above, we aim to define an FO sentence that decides recoverability of a database.

Given a set of acyclic+linear dependencies Σ, where $\langle \tau_1, \ldots, \tau_h \rangle$ is a topological order of Σ, and an atom $p(\mathbf{t})$, we define the formula $\Psi^{rec}(p(\mathbf{t}), \Sigma)$ as follows: (i) if $\Sigma = \emptyset$, then $\Psi^{rec}(p(\mathbf{t}), \Sigma) = true$; (ii) if $\Sigma \neq \emptyset$, then $\Psi^{rec}(p(\mathbf{t}), \Sigma)$ is the formula:

$$\bigwedge_{\substack{\tau_i \in \Sigma \wedge \\ PB(\tau_i) = \{p(\mathbf{t}')\}}} \forall \mathbf{x} \Big(body(\tau_i) \wedge \mathbf{t} = \mathbf{t}' \rightarrow \Big(\bigvee_{\exists \mathbf{y}\, \gamma \in CQ(head(\tau_i))} \exists \mathbf{y} \Big(\gamma \wedge \bigwedge_{\beta \in predAt(\gamma)} \Psi^{rec}(\beta, \{\tau_{i+1}, \ldots, \tau_h\}) \Big) \Big) \Big)$$

where $PB(\tau_i) = predAt(body(\tau_i))$, and $\mathbf{x}$ and $\mathbf{t}'$ are, respectively, all the variables and all the terms occurring in $body(\tau_i)$. Note that each recursive call removes the first dependency from the topological order of Σ: if the remaining set is empty, then the function returns true, otherwise we look for dependencies in Σ whose body's predicate is p. For each such dependency τ_i, we impose the equality between the terms of $p(\mathbf{t})$ and $body(\tau_i)$[4] and under such a "binding" we check if, for at least one disjunct q in the head, both q and the recursive subformula taking as input β and the remaining dependencies are true.

Thus, the following holds:

Lemma 16. *Let Σ be a set of acyclic+linear dependencies, let α be an atom, and let σ be a substitution of the variables occurring in α with constants. Then, for every database $\mathcal{D}$, $\sigma(\alpha)$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$ iff $\sigma(\alpha) \in \mathcal{D}$ and $\sigma(\Psi^{rec}(\alpha, \Sigma))$ evaluates to true in $\mathcal{D}$.*

Then, we define the formula $\Psi^{rec}(\Sigma)$ as follows:

$$\Psi^{rec}(\Sigma) = \bigwedge_{p \in pred(\Sigma)} \forall \mathbf{x} \left(\overset{*}{p}(\mathbf{x}) \rightarrow p(\mathbf{x}) \wedge \Psi^{rec}(p(\mathbf{x}), \Sigma) \right)$$

[4]Note again that $\mathbf{t} = \mathbf{t}'$ could also contain an equality between distinct constants, which would simply make false the evaluation of the premise.

Proposition 30. *RECOVERABILITY is in AC^0 in data complexity in the case of acyclic+linear dependencies.*

Proof. From Lemma 16 it immediately follows that $\mathcal{D}'$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$ iff $\Psi^{rec}(\Sigma)$ evaluates to true in $\mathcal{D} \cup aux(\mathcal{D}')$. Now, since $\Psi^{rec}(\Sigma)$ is an FO sentence, the thesis follows. ∎

Example 41. For modeling a social network we use three predicates, namely U for users (accepting a user ID, a nickname, and a registration date), P for posts (accepting a post ID and an author), and L for the "like" reaction (accepting a user ID and a post ID). Only users can like posts, only posts can be liked, and posts must be authored by users. Moreover, we know that the user u_1 (with nick yoda and registration date d_1) likes both posts p_1, created by herself, and p_2, whose author u_2 is not known to be a user. We thus consider the following database and acyclic+linear set of dependencies:

$$\begin{aligned}\mathcal{D} &= \{U(\mathsf{u}_1, \mathsf{yoda}, \mathsf{d}_1), P(\mathsf{p}_1, \mathsf{u}_1), P(\mathsf{p}_2, \mathsf{u}_2), L(\mathsf{u}_1, \mathsf{p}_1), L(\mathsf{u}_1, \mathsf{p}_2)\},\\ \Sigma &= \{\forall u, p\,(L(u,p) \rightarrow \exists n, r, a\,(U(u,n,r) \wedge P(p,a))),\\ &\qquad \forall p, a\,(P(p,a) \rightarrow \exists n, r\,U(a,n,r))\}.\end{aligned}$$

Then, the sentence $\Psi^{rec}(\Sigma)$ is equal to:

$$\begin{aligned}&\forall u, p\,(\overset{*}{L}(u,p) \rightarrow L(u,p) \wedge \exists n, r, a\,(U(u,n,r) \wedge P(p,a) \wedge\\ &\qquad \forall p', a'(P(p',a') \wedge p = p' \wedge a = a' \rightarrow \exists n', r'\,U(a,n',r')))) \wedge\\ &\forall p, a\,(\overset{*}{P}(p,a) \rightarrow P(p,a) \wedge \exists n, r\,U(a,n,r)).\end{aligned}$$

which can be simplified in:

$$\begin{aligned}&\forall u, p\ (\overset{*}{L}(u,p) \rightarrow L(u,p) \wedge \exists n, r, a\,(U(u,n,r) \wedge P(p,a) \wedge \exists n', r'\,U(a,n',r'))) \wedge\\ &\forall p, a\ (\overset{*}{P}(p,a) \rightarrow P(p,a) \wedge \exists n, r\,U(a,n,r)).\end{aligned}$$

Let us now consider two subsets of $\mathcal{D}$, namely $\mathcal{D}' = \{L(\mathsf{u}_1, \mathsf{p}_1)\}$ and $\mathcal{D}'' = \{L(\mathsf{u}_1, \mathsf{p}_2)\}$. One can verify that the evaluation of $\Psi^{rec}(\Sigma)$ returns true in $\mathcal{D} \cup aux(\mathcal{D}')$ and false in $\mathcal{D} \cup aux(\mathcal{D}'')$. Indeed, we have that the set $\mathcal{D}'$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$, while $\mathcal{D}''$ is not. ▽

8.4 Repair checking

In this section, we study the data complexity of repair checking in the classes of dependencies previously mentioned.

Lower bounds

We remark that all the lower bounds shown in this section also hold if the dependencies are restricted to be single-head and without inequalities in the head.

First, we focus on the lower bounds of the problem of checking if a given database is a repair. We prove the repair checking problem to be PTIME-hard in the case when such dependencies are linear+sticky.

Theorem 17. *There exists a set of linear+sticky dependencies for which* Rep-Checking *is* PTime*-hard in data complexity.*

Proof. We prove the thesis by showing a reduction from Horn-CNF-SAT.

Let ϕ be a set of n ground Horn rules. We assume w.l.o.g. that there is at least one rule without head in ϕ (i.e. a clause with only negated variables). Let ϕ' be obtained from ϕ by slightly modifying rules without heads as follows: every rule without head r is replaced by the rule with the same body as r and with the new variable u in the head; moreover, if ϕ contains only one rule without head, then we add the rule $u \rightarrow u$ to ϕ' (so ϕ' always contains at least two rules with u in the head). It is immediate to verify that ϕ is unsatisfiable iff u belongs to all the models (and hence to the minimal model) of ϕ'.

Now, we associate an id r_i to every Horn rule in ϕ', with r_1 associated to a rule having u in the head. Then, let $\phi'(x)$ be the rules of ϕ' having the variable x in their head, let $PV(\phi')$ be the set of propositional variables occurring in ϕ' and, for every rule r_i, let $BV(r_i)$ be the set of variables in the body of r_i. We define $\mathcal{D}$ as the set of facts:

$$\bigcup_{x \in PV(\phi')} \{H(r_{i_1}, x, 1, 2), H(r_{i_2}, x, 2, 3), \ldots, H(r_{i_h}, x, h, 1) \mid h = |\phi'(x)|\} \cup \bigcup_{r_i \in \phi'} \{B(r_i, x) \mid x \in BV(r_i)\}$$

Finally, we define the following set of linear+sticky dependencies Σ:

$$\begin{aligned} \forall x, y, z, w\ (H(x, y, z, w) &\rightarrow \exists v\, B(x, v)) \\ \forall x, y, z, w\ (H(x, y, z, w) &\rightarrow \exists v, t\, H(v, y, w, t)) \\ \forall x, y\ (B(x, y) &\rightarrow \exists z, w, v\, H(z, y, w, v)) \\ \forall x, y, z, w\ (H(x, y, z, w) &\rightarrow H(r_1, u, 1, 2)) \\ \forall x, y\ (B(x, y) &\rightarrow H(r_1, u, 1, 2)) \end{aligned}$$

Now, the (unique) repair of $\langle \Sigma, \mathcal{D} \rangle$ is the set of facts built starting from $\mathcal{D}$ and deterministically removing some atoms according to the dependencies of Σ. Observe that:

- because of the first dependency of Σ, if for a rule there are no more body atoms, then the head atom of that rule is deleted;
- because of the second dependency, if a head atom is deleted, then all the head atoms with that variable are deleted;
- because of the third dependency, if there are no head atoms for a variable, then all the body atoms for that variable are deleted;
- because of the fourth and fifth dependencies, if the atom $H(r, u, 1, 2)$ is deleted, then all the atoms are deleted.

Therefore, it can be verified that for every variable x that belongs to the minimal model of ϕ', no fact of the form $H(_, x, _, _)$ or $B(x, _)$ belong to the repair of $\langle \Sigma, \mathcal{D} \rangle$. Consequently, the minimal model of ϕ' contains u (i.e. ϕ is unsatisfiable) iff the fact $H(r_1, u, 1, 2)$ does not belong to the repair of $\langle \Sigma, \mathcal{D} \rangle$, which implies (because

of the fourth and fifth rules of Σ) that ϕ is unsatisfiable iff the empty database is the repair of $\langle \Sigma, \mathcal{D} \rangle$. ∎

Then, we extend the PTIME lower bound shown in [2, Theorem 5] for full dependencies to the case of full+guarded dependencies.

Theorem 18. *There exists a set of full+guarded dependencies for which* REP-CHECKING *is* PTIME*-hard in data complexity.*

Proof. We prove the thesis using a modification of the reduction from HORN-3-SAT shown in the proof of [2, Theorem 5] for the case of full TGDs.

Let ϕ be a Horn 3-CNF formula. We denote by $\mathit{clauses}(\phi)$ the set of clauses of ϕ, and by $PV(\phi)$ the set of propositional variables of ϕ. Let $\mathcal{D}$ and $\mathcal{D}'$ be the following sets of facts:

$$\begin{aligned}\mathcal{D}' = &\{U(x,y) \mid x \in \mathit{clauses}(\phi), y \in PV(\phi)\} \cup \\ &\{P(x,y,z) \mid x, y \rightarrow z \in \mathit{clauses}(\phi)\} \cup \\ &\{N(x,y,z) \mid x, y, z \rightarrow \bot \in \mathit{clauses}(\phi)\} \\ \mathcal{D} = &\mathcal{D}' \cup \{T(x), V(x) \mid x \in PV(\phi)\}\end{aligned}$$

It is crucial to observe that both $\mathcal{D}$ and $\mathcal{D}'$ contain a fact $U(x,y)$ if the x is a propositional variable belonging to a *unit clause* of ϕ and y is *any* variable of ϕ. Then, let Σ be a set consisting of the following full+guarded dependencies:

$$\begin{aligned}\forall x, y, z\ (N(x,y,z) \wedge T(x) \wedge T(y) \wedge T(z) \rightarrow F) \\ \forall x, y, z\ (P(x,y,z) \wedge T(x) \wedge T(y) \rightarrow T(z)) \\ \forall x, y\ (V(y) \wedge U(x,y) \rightarrow T(x)) \\ \forall x\ (T(x) \rightarrow V(x))\end{aligned}$$

We now show that ϕ is unsatisfiable iff $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$.

($\Rightarrow$) ϕ is unsatisfiable only if $\mathcal{D}$ violates at least one of the first three dependencies. Then, starting from $\mathcal{D}$, we can build a repair of $\langle \Sigma, \mathcal{D} \rangle$ as follows.

1. For each violation of the first or second dependency, we choose to delete one T-fact in the image of the body of such dependency which causes the violation. Since this is not sufficient for achieving consistency (otherwise ϕ would be satisfiable), we enter a loop, and eventually the third dependency will be violated. Then, go to the next step.
2. If the third dependency is violated, we choose to delete all the V-facts. Then, because of the fourth dependency, we also have to remove all the T-facts.

The obtained repair coincides with $\mathcal{D}'$.

($\Leftarrow$) On the other hand, if ϕ is satisfiable, then $\mathcal{D}'$ is not a repair of $\langle \Sigma, \mathcal{D} \rangle$ because the database $\mathcal{D}'' = \mathcal{D}' \cup \{T(x), V(x) \mid x \in MM(\phi)\}$, where $MM(\phi)$ is the minimal model of ϕ, satisfies Σ. ∎

Furthermore, the results provided in [51] imply that, in the general case, repair checking is coNP-hard. We establish the same lower bound for the case of guarded+sticky dependencies.

Theorem 19. *There exists a set of guarded+sticky dependencies for which* Rep-Checking *is coNP-hard in data complexity.*

Proof. We prove the thesis by showing a reduction from CNF-SAT. The reduction is obtained by modifying the reduction from 2-QBF$^{\forall}$ shown in Theorem 4.7 of [51].

Let Σ be a set consisting of the following guarded+sticky dependencies:

$$\forall x, y, z, w\ (R(x, y, z, w) \rightarrow \exists x', y', z'\, R(x', y', z', z))$$
$$\forall x, y, z, y', z'\ (R(x, 0, y, z) \wedge R(x, 1, y', z') \wedge R_{aux}(x, y, z, y', z') \rightarrow U(x, y, z, y', z'))$$

Now let $\phi = \psi_1 \wedge \ldots \wedge \psi_n$ be a CNF formula, where every ψ_i is a clause. Let $PV(\phi)$ be the set of propositional variables occurring in ϕ. Moreover, let $\mathcal{D}'$ and $\mathcal{D}''$ be the following sets of facts:

$$\begin{aligned}\mathcal{D}' = {} & \{R(a, 1, i, (i \bmod n) + 1) \mid a \text{ occurs positively in } \psi_i\} \cup \\ & \{R(a, 0, i, (i \bmod n) + 1) \mid a \text{ occurs negatively in } \psi_i\} \\ \mathcal{D}'' = {} & \{R_{aux}(v, i, j, (i \bmod n) + 1, (j \bmod n) + 1) \mid v \in PV(\phi) \text{ and } i, j \in \{1, ..., n\}\}\end{aligned}$$

and let $\mathcal{D} = \mathcal{D}' \cup \mathcal{D}''$. We prove that $\mathcal{D}''$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$ iff ϕ is unsatisfiable.

First, it is immediate to see that $\mathcal{D}''$ satisfies Σ. Now, due to the first dependency of Σ, if there exists $\mathcal{D}'''$ such that $\mathcal{D}'' \subset \mathcal{D}''' \subseteq \mathcal{D}$ and $\mathcal{D}'''$ satisfies Σ, then for every conjunct ψ_i there must exist at least one fact from R of the form $R(_, _, i, _)$ in $\mathcal{D}'''$. But, due to the second dependency of Σ, this is possible iff ϕ is satisfiable. Consequently, $\mathcal{D}''$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$ iff ϕ is unsatisfiable. ∎

Upper bounds

As shown in [2, Proposition 4], in the general case (actually, for «every finite set of first-order constraints»), Rep-Checking is in coNP. We show that the problem can be solved in polynomial time if the set of dependencies is either linear or full, extending the same complexity result proved in [110, 114] for less expressive dependencies. Moreover, we show that it is in AC^0 for acyclic sets of dependencies, as well as for full+linear and full+sticky dependencies.

In order to exploit the previous results for recoverability, we start by noticing the following property (whose proof is straightforward):

Proposition 31. *Let Σ be a set of dependencies, and let $\mathcal{D}$ and $\mathcal{D}'$ be two databases such that $\mathcal{D}' \subseteq \mathcal{D}$. Then, $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$ iff $\mathcal{D}'$ satisfies Σ and, for every $\alpha \in \mathcal{D} \setminus \mathcal{D}'$, $\mathcal{D}' \cup \{\alpha\}$ is not recoverable from $\langle \Sigma, \mathcal{D} \rangle$.*

Consequently, whenever deciding recoverability is in PTime in data complexity, also repair checking is in PTime in data complexity.

Thus, from Lemma 12 and Lemma 15 we can prove the following result.

Theorem 20. *Rep-Checking is in PTime in data complexity in the case of either linear or full dependencies.*

We now focus on acyclic dependencies, for which we prove what follows.

Proposition 32. *Let Σ be a set of acyclic dependencies, and let $\mathcal{D}$ and $\mathcal{D}'$ be two databases such that $\mathcal{D}' \subseteq \mathcal{D}$. Then, $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$ iff $\mathcal{D}'$ satisfies Σ and, for every $\alpha \in \mathcal{D} \setminus \mathcal{D}'$, $\mathcal{D}' \cup \{\alpha\}$ does not satisfy Σ.*

Proof. We prove that, if $\mathcal{D}'$ is not a repair of $\langle \Sigma, \mathcal{D} \rangle$ and $\mathcal{D}' \subseteq \mathcal{D}$ and $\mathcal{D}'$ satisfies Σ, then there exists $\alpha \in \mathcal{D} \setminus \mathcal{D}'$ such that $\mathcal{D}' \cup \{\alpha\}$ satisfies Σ (the other direction of the proof is straightforward). So let us assume that $\mathcal{D}'$ is not a repair of $\langle \Sigma, \mathcal{D} \rangle$ and $\mathcal{D}' \subseteq \mathcal{D}$ and $\mathcal{D}'$ satisfies Σ.

First, observe that, since $\mathcal{D}'$ is not a repair of $\langle \Sigma, \mathcal{D} \rangle$, then there must exist an $\alpha \in \mathcal{D} \setminus \mathcal{D}'$ such that $\mathcal{D}' \cup \{\alpha\}$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$.

Then, let us stratify the set of predicates according to the dependencies: layer 1 contains predicates not occurring in the body of any dependency; layer i contains predicates occurring only in the body of dependencies whose head contains only predicates of layers lower than i. Moreover, we say that a dependency τ is of layer i if no predicate of layer greater than i occurs in the head of τ.

Now, let i be the minimum layer such that there exists $\alpha \in \mathcal{D} \setminus \mathcal{D}'$ such that $\mathcal{D}' \cup \{\alpha\}$ is recoverable from $\langle \Sigma, \mathcal{D} \rangle$ and the predicate of α belongs to layer i. Let α be such a fact whose predicate is of layer i and suppose that $\mathcal{D}' \cup \{\alpha\}$ does not satisfy Σ. Then, since $\mathcal{D}'$ satisfies Σ, it follows that $\mathcal{D}' \cup \{\alpha\}$ violates a dependency of layer $i-1$. Now observe that, for every layer j such that $j < i$, and for every $\beta \in \mathcal{D} \setminus \mathcal{D}'$ such that $\mathcal{D}' \cup \{\beta\}$ does not satisfy Σ and the predicate of β belongs to layer j, $\mathcal{D}' \cup \{\beta\}$ is not recoverable from $\langle \Sigma, \mathcal{D} \rangle$. This implies that $\mathcal{D}' \cup \{\alpha\}$ is not recoverable from $\langle \Sigma, \mathcal{D} \rangle$, giving rise to a contradiction. Thus, $\mathcal{D}' \cup \{\alpha\}$ satisfies Σ. ∎

We now show how to decide repair checking by means of an FO sentence. As done in Section 8.3, we use an auxiliary predicate $\overset{*}{p}$ for every $p \in \mathit{pred}(\Sigma)$.

First, given any set Φ of domain-independent FO sentences, we define the formula $\Psi^{sat}(\Phi)$ as follows:

$$\Psi^{sat}(\Phi) = \bigwedge_{\phi \in \Phi} \mathit{aux}(\phi)$$

Clearly, the $\Psi^{sat}(\Phi)$ evaluates to true in a database $\mathit{aux}(\mathcal{D})$ iff each $\phi \in \Phi$ does. In particular, given a set of dependencies Σ, the evaluation of $\Psi^{sat}(\Sigma)$ over a database $\mathit{aux}(\mathcal{D})$ is true iff the latter satisfies Σ.

Then, we define the auxiliary formula $\Psi^{sat}(\Sigma, \overset{*}{p}(\mathbf{x}))$, where $\overset{*}{p}(\mathbf{x})$ is a predicate atom: it is obtained from $\Psi^{sat}(\Sigma)$ by replacing every atom $\overset{*}{p}(\mathbf{t})$ with the subformula $(\overset{*}{p}(\mathbf{t}) \vee \mathbf{t} = \mathbf{x})$.

Definition 37. [Repair checking sentence 1] Let Σ be a set of acyclic dependencies. We define the formula $\Psi^{rc}(\Sigma)$ as follows:

$$\Psi^{rc}(\Sigma) = \Psi^{sat}(\Sigma) \wedge \bigwedge_{p \in \mathit{pred}(\Sigma)} \forall \mathbf{x} \left(p(\mathbf{x}) \wedge \neg \overset{*}{p}(\mathbf{x}) \rightarrow \neg \Psi^{sat}(\Sigma, \overset{*}{p}(\mathbf{x})) \right)$$

where $\mathbf{x} = x_1, \ldots, x_m$ if m is the arity of p.

Based on this definition and Proposition 32, it is possible to prove the following property.

Theorem 21. *Let Σ be a set of acyclic dependencies, and let $\mathcal{D}$ and $\mathcal{D}'$ be two databases. Then, the sentence $\Psi^{rc}(\Sigma)$ evaluates to true in $\mathcal{D} \cup \mathit{aux}(\mathcal{D}')$ iff $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$.*

Proof. First, it is immediate to verify that, for every instantiation σ of $p(\mathbf{x})$, the sentence $\Psi^{sat}(\Sigma, \sigma(\overset{*}{p}(\mathbf{x})))$ evaluates to true in $aux(\mathcal{D}')$ iff $\Psi^{sat}(\Sigma)$ evaluates to true in $aux(\mathcal{D}') \cup \{\sigma(\overset{*}{p}(\mathbf{x}))\}$. Then, from Proposition 32 and from the fact that $\Psi^{sat}(\Sigma)$ evaluates to true in $aux(\mathcal{D}')$ iff $\mathcal{D}'$ satisfies Σ, it follows that $\Psi^{rc}(\Sigma)$ evaluates to true in $\mathcal{D} \cup aux(\mathcal{D}')$ iff $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$. ∎

Corollary 7. *Rep-Checking is in AC^0 in data complexity in the case of acyclic dependencies.*

Example 42. IUCN is an organization studying the conservation status of species worldwide. If a taxon is a species (predicate S), and it has been described (predicate D) by IUCN, then we know its conservation status (predicate C). Moreover, our dataset contains the fact that both species *Mammuthus primigenius* and *Elephas maximus* have been described by IUCN, and that the first species is extinct. This situation can be modeled through the following database and acyclic set of dependencies:

$$\mathcal{D} = \{\ S(\mathsf{e}), D(\mathsf{e}, \mathsf{iucn}), S(\mathsf{m}), D(\mathsf{m}, \mathsf{iucn}), C(\mathsf{m}, \mathsf{ex})\ \},$$
$$\Sigma = \{\ \forall x\,(S(x) \wedge D(x, \mathsf{iucn}) \rightarrow \exists y\, C(x, y))\ \}.$$

Then, the sentence $\Psi^{rc}(\Sigma)$ is equivalent to:

$$\begin{aligned}
&\neg\exists x\,(\overset{*}{S}(x) \wedge \overset{*}{D}(x, \mathsf{iucn}) \wedge \neg\exists y\, \overset{*}{C}(x, y)) \wedge \\
&\quad \forall x\,(S(x) \wedge \neg\overset{*}{S}(x) \rightarrow \exists x'\,((\overset{*}{S}(x') \vee x' = x) \wedge \overset{*}{D}(x', \mathsf{iucn}) \wedge \neg\exists y'\, \overset{*}{C}(x', y'))) \wedge \\
&\forall x, y\,(D(x, y) \wedge \neg\overset{*}{D}(x, y) \rightarrow \exists x'\,(\overset{*}{S}(x') \wedge (\overset{*}{D}(x', \mathsf{iucn}) \vee (x' = x \wedge \mathsf{iucn} = y)) \wedge \\
&\qquad\qquad \neg\exists y'\, \overset{*}{C}(x', y'))) \wedge \\
&\forall x, y\,(C(x, y) \wedge \neg\overset{*}{C}(x, y) \rightarrow \exists x'\,(\overset{*}{S}(x') \wedge \overset{*}{D}(x', \mathsf{iucn}) \wedge \\
&\qquad\qquad \neg\exists y'\,(\overset{*}{C}(x', y') \vee (x' = x \wedge y' = y)))).
\end{aligned}$$

One can verify that $\mathsf{rep}_\Sigma(\mathcal{D}) = \{\mathcal{D} \setminus \{S(\mathsf{e})\}, \mathcal{D} \setminus \{D(\mathsf{e}, \mathsf{iucn})\}\}$ and that these are the only two possible subsets of $\mathcal{D}$ such that $\Psi^{rc}(\Sigma)$ evaluates to true in $\mathcal{D} \cup aux(\mathcal{D}')$. ▽

Finally, we address the cases of full+sticky and full+linear dependencies. To this aim, we exploit the formulas for checking recoverability in case Σ is a set of CQ-FO-rewritable dependencies. Indeed, we define the FO sentence $\Phi^{rc}(\Sigma)$ as follows:

Definition 38. [Repair checking sentence 2] Given a CQ-FO-rewritable set of dependencies Σ, we define $\Phi^{rc}(\Sigma)$ as the FO sentence:

$$\Phi^{rc}(\Sigma) = \Phi^{rec}(\Sigma) \wedge \bigwedge_{p \in pred(\Sigma)} \neg\exists \mathbf{y}\,(p(\mathbf{y}) \wedge \neg\overset{*}{p}(\mathbf{y}) \wedge \Phi^{rec}(\{\overset{*}{p}(\mathbf{y})\}, \Sigma))$$

The intuition behind the formula is straightforward: we check if the database is recoverable (which is a necessary condition for it to be a repair) and no further atom not already present in it can be added while preserving its recoverability.

It is now possible to show that $\Phi^{rc}(\Sigma)$ constitutes a sound and complete FO-rewriting for the repair checking problem, i.e. such a formula is evaluated to true over the database $\mathcal{D} \cup aux(\mathcal{D}')$ iff $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$. Therefore, the following property holds.

Theorem 22. *Rep-Checking is in AC^0 in data complexity in the case of CQ-FO-rewritable sets of full dependencies.*

Proof. First, we state the following property, whose proof immediately follows from the definition of chase and the definition of repair.

Lemma 17. *Let Σ be a set of full dependencies, and let $\mathcal{D}$ and $\mathcal{D}'$ be two databases. Then, $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$ iff $Chase(\mathcal{D}', \Sigma) \subseteq \mathcal{D}$ and, for every fact $\alpha \in \mathcal{D} \setminus \mathcal{D}'$, $Chase(\mathcal{D}' \cup \{\alpha\}, \Sigma) \not\subseteq \mathcal{D}$.*

Now, given a CQ-FO-rewritable set of full dependencies Σ, we can encode the conditions of the above lemma in terms of an FO sentence. Indeed, Lemma 14 and Lemma 17 imply that, for every set of dependencies Σ that is full and CQ-FO-rewritable, $\Phi^{rc}(\Sigma)$ evaluates to true in $\mathcal{D} \cup aux(\mathcal{D}')$ iff $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$. Thus, repair checking is FO-rewritable for every Σ that is full and CQ-FO-rewritable and, consequently, the problem is in AC^0. ∎

As an immediate consequence of Theorem 22 and Proposition 29, we obtain what follows.

Corollary 8. *Rep-Checking is in AC^0 in data complexity in the case of full+linear or full+sticky dependencies.*

8.5 IAR-entailment

In this section we focus on the problem of IAR-entailment of $\mathrm{BUCQ}^{\neq}$s.

Lower bounds

First, we are able to show that instance checking is already Π_2^p-hard for the class of guarded+sticky dependencies.

Theorem 23. *There exist a set of guarded+sticky dependencies and a fact for which Instance-Checking is Π_2^p-hard in data complexity.*

Proof. We prove the thesis by showing a reduction from 2-$\mathrm{QBF}^{\forall}$ that is obtained by slightly modifying the reduction, from the same problem, shown in Theorem 4.7 of [51] (as well as the reduction shown in the proof of Theorem 19).

Let Σ be a set consisting of the following guarded+sticky dependencies:

$$\forall x, y, z, w \; (R(x, y, z, w) \rightarrow \exists x', y', z' \, R(x', y', z', z))$$
$$\forall x, y, z, y', z' \; (R(x, 0, y, z) \wedge R(x, 1, y', z') \wedge R_{aux}(x, y, z, y', z') \rightarrow U(x, y, z, y', z'))$$

Now let $\phi = \forall \mathbf{x} \, \exists \mathbf{y} \, (\psi_1 \wedge \ldots \wedge \psi_n)$ be a QBF, where every ψ_i is a clause over the propositional variables $\mathbf{x}, \mathbf{y}$. Let $PV(\phi) = \mathbf{x} \cup \mathbf{y}$. Moreover, let $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$ and $\mathcal{D}$ be the following sets of facts:

$$\begin{aligned}
\mathcal{D}_1 &= \{R(z, 1, i, (i \bmod n) + 1) \mid z \text{ occurs positively in } \psi_i\} \cup \\
&\quad \{R(z, 0, i, (i \bmod n) + 1) \mid z \text{ occurs negatively in } \psi_i\} \\
\mathcal{D}_2 &= \{R_{aux}(v, i, j, (i \bmod n) + 1, (j \bmod n) + 1) \mid v \in PV(\phi) \text{ and } i, j \in \{1, ..., n\}\} \\
\mathcal{D}_3 &= \{R(x, 1, 0, 0), R(x, 0, 0, 0) \mid x \in \mathbf{x}\} \\
\mathcal{D} &= \mathcal{D}_1 \cup \mathcal{D}_2 \cup \mathcal{D}_3 \cup \{R(a, a, 1, a)\}
\end{aligned}$$

We prove that $\langle \Sigma, \mathcal{D} \rangle \models^{c}_{\text{AR}} R(a,a,1,a)$ iff ϕ is valid.

First, for every interpretation $I_{\mathbf{x}}$ of $\mathbf{x}$, let $\mathcal{D}(I_{\mathbf{x}}) = \{R(x,1,0,0) \mid I_{\mathbf{x}}(x) = 1\} \cup \{R(x,0,0,0) \mid I_{\mathbf{x}}(x) = 0\}$ (notice that $\mathcal{D}(I_{\mathbf{x}}) \subset \mathcal{D}_3$).

Now let us assume that ϕ is not valid, and let $I_{\mathbf{x}}$ be an interpretation of $\mathbf{x}$ such that there exists no interpretation $I_{\mathbf{y}}$ of $\mathbf{y}$ such that $I_{\mathbf{x}} \cup I_{\mathbf{y}}$ satisfies $\psi_1 \wedge \ldots \wedge \psi_n$. We prove that $\mathcal{D}_2 \cup \mathcal{D}(I_{\mathbf{x}})$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$. Indeed, if a fact from $\mathcal{D}_1 \cup \{R(a,a,1,a)\}$ is added to such a set, then consistency would be possible only by adding, for every conjunct ψ_i, at least one fact of the form $R(_,_,i,_)$ in $\mathcal{D}_1$. But, due to due to the second dependency of Σ, this is possible iff ϕ is valid. Consequently, $\mathcal{D}_2 \cup \mathcal{D}(I_{\mathbf{x}})$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$, which implies that $\langle \Sigma, \mathcal{D} \rangle$ does not AR-entail $R(a,a,1,a)$.

Conversely, assume that ϕ is valid. Let $I_{\mathbf{x}}$ be any interpretation of $\mathbf{x}$. Then, there exists an interpretation $I_{\mathbf{y}}$ of $\mathbf{y}$ such that $I_{\mathbf{x}} \cup I_{\mathbf{y}}$ satisfies $\psi_1 \wedge \ldots \wedge \psi_n$. Now let $\mathcal{D}(I_{\mathbf{y}}) = \{R(y,z,w,v) \mid y \in \mathbf{y} \wedge z = I_{\mathbf{y}}(y) \wedge R(y,z,w,v) \in \mathcal{D}_1\}$. It is immediate to verify that $\mathcal{D}_2 \cup \mathcal{D}(I_{\mathbf{y}})$ satisfies Σ. Now, suppose $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$ and suppose $R(a,a,1,a) \notin \mathcal{D}'$. This implies that all the facts of $\mathcal{D}_1$ of the form $R(_,_,n,1)$ do not belong to $\mathcal{D}'$ (because $R(a,a,1,a)$ can only violate the first dependency of Σ), which in turn implies (again due to the first dependency of Σ) that all the facts of $\mathcal{D}_1$ of the form $R(_,_,n-1,n)$ do not belong to $\mathcal{D}'$, and so on: this iteratively proves that *all* the facts of $\mathcal{D}_1$ do not belong to $\mathcal{D}'$. Therefore, $R(a,a,1,a) \notin \mathcal{D}'$ implies that $\mathcal{D}' \cap \mathcal{D}_1 = \emptyset$, that is, there is no fact of the form $R(y,_,_,_)$ with $y \in \mathbf{y}$ in $\mathcal{D}'$. But this immediately implies that $\mathcal{D}' \cup \mathcal{D}(I_{\mathbf{y}})$ satisfies Σ, thus contradicting the hypothesis that $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$. Consequently, $R(a,a,1,a)$ belongs to every repair of $\langle \Sigma, \mathcal{D} \rangle$, and therefore $\langle \Sigma, \mathcal{D} \rangle \models^{c}_{\text{AR}} R(a,a,1,a)$. ∎

We can also prove a coNP lower bound for instance checking in the case of acyclic+guarded+sticky dependencies and in the case of full+guarded dependencies.

Theorem 24. *There exist a set of acyclic+guarded+sticky dependencies and a fact for which* Instance-Checking *is coNP-hard in data complexity.*

Proof. The proof is obtained through a reduction from 3-SAT. Let Σ be a set consisting of the following acyclic+guarded+sticky dependencies:

$$
\begin{aligned}
& \forall x\ (V(x,1) \wedge V(x,0) \rightarrow U_1(x)) \\
\forall x, y_1, y_2, y_3, z_1, z_2, z_3\ & (C_1(x, y_1, z_1, y_2, z_2, y_3, z_3) \wedge \\
& \quad V(y_1, z_1) \wedge V(y_2, z_2) \wedge V(y_3, z_3) \rightarrow U_2(y_1, z_1, y_2, z_2, y_3, z_3)) \\
& \ldots \\
\forall x, y_1, y_2, y_3, z_1, z_2, z_3\ & (C_7(x, y_1, z_1, y_2, z_2, y_3, z_3) \wedge \\
& \quad V(y_1, z_1) \wedge V(y_2, z_2) \wedge V(y_3, z_3) \rightarrow U_2(y_1, z_1, y_2, z_2, y_3, z_3)) \\
\forall x, y_1, y_2, y_3, y_4, y_5, y_6\ & (C_1(x, y_1, y_2, y_3, y_4, y_5, y_6) \rightarrow \\
& \quad \exists w_1, w_2, w_3, w_4, w_5, w_6\ C_2(x, w_1, w_2, w_3, w_4, w_5, w_6)) \\
& \ldots \\
\forall x, y_1, y_2, y_3, y_4, y_5, y_6\ & (C_6(x, y_1, y_2, y_3, y_4, y_5, y_6) \rightarrow \\
& \quad \exists w_1, w_2, w_3, w_4, w_5, w_6\ C_7(x, w_1, w_2, w_3, w_4, w_5, w_6)) \\
& U \rightarrow \exists x, y_1, y_2, y_3, y_4, y_5, y_6\ C_1(x, y_1, y_2, y_3, y_4, y_5, y_6)
\end{aligned}
$$

Given a 3-CNF formula ϕ, we define $\mathcal{D}$ as be the database containing: $V(a,0), V(a,1)$ for each variable a occurring in ϕ, seven facts $C_1, \ldots, C_7$ for each clause of ϕ (each

such fact represents an evaluation of the three variables of the clause that make the clause true), and the fact U. E.g. if the i-th clause is $\neg a \vee \neg b \vee c$, then:

$$\begin{aligned}\mathcal{D} = \{\ &\ldots, V(a,0), V(a,1), V(b,0), V(b,1), V(c,0), V(c,1),\\ &C_1(i,a,0,b,0,c,0), C_2(i,a,0,b,0,c,1), C_3(i,a,0,b,1,c,0), C_4(i,a,0,b,1,c,1),\\ &C_5(i,a,1,b,0,c,0), C_6(i,a,1,b,0,c,1), C_7(i,a,1,b,1,c,1), \ldots, U\ \}\end{aligned}$$

We prove that the fact U belongs to all the repairs of $\langle \Sigma, \mathcal{D} \rangle$ iff ϕ is unsatisfiable.

In fact, if ϕ is unsatisfiable, then there is no repair of $\langle \Sigma, \mathcal{D} \rangle$ without any fact of C_1, consequently U belongs to all the repairs of $\langle \Sigma, \mathcal{D} \rangle$. Conversely, if ϕ is satisfiable, then there is a repair (where the extension of V corresponds to the interpretation satisfying ϕ) that for each clause does not contain at least one of the 7 facts $C_1, \ldots, C_7$ representing the clause. Due to the dependencies between C_i and C_{i+1}, this implies that there is a repair that for each clause does not contain any fact C_1, and therefore (due to the last dependency) it does not contain the fact U. ∎

Theorem 25. *There exist a set of full+guarded dependencies and a fact for which* Instance-Checking *is coNP-hard in data complexity.*

Proof. The proof is by reduction from 3-SAT. We define the following set Σ of full+guarded dependencies:

$$\begin{gathered}\forall x_1, x_2, x_3, v_1, v_2, v_3, y, z\ (S(z) \wedge C(y, z, x_1, v_1, x_2, v_2, x_3, v_3) \wedge V(x_1, v_1) \rightarrow S(y))\\ \forall x_1, x_2, x_3, v_1, v_2, v_3, y, z\ (S(z) \wedge C(y, z, x_1, v_1, x_2, v_2, x_3, v_3) \wedge V(x_2, v_2) \rightarrow S(y))\\ \forall x_1, x_2, x_3, v_1, v_2, v_3, y, z\ (S(z) \wedge C(y, z, x_1, v_1, x_2, v_2, x_3, v_3) \wedge V(x_3, v_3) \rightarrow S(y))\\ \forall x\ (V(x, 0) \wedge V(x, 1) \rightarrow U)\end{gathered}$$

Given a 3-CNF formula ϕ with m clauses, in the database $\mathcal{D}$, we represent every clause of ϕ with a fact C: e.g. if the i-th clause of ϕ is $a \vee \neg b \vee c$, we add the fact $C(i-1, i, a, 1, b, 0, c, 1)$.

Moreover, the database contains the facts $V(p,0), V(p,1)$ for every propositional variable p, and the facts $\{S(1), S(2), \ldots, S(m)\}$.

We prove that ϕ is unsatisfiable iff the fact $S(m)$ belongs to all the repairs of $\langle \Sigma, \mathcal{D} \rangle$.

First, if ϕ is satisfiable, then let P be the set of propositional variables occurring in ϕ, let I be an interpretation (subset of P) satisfying ϕ, and let $\mathcal{D}'$ be the following subset of $\mathcal{D}$:

$$\begin{aligned}\mathcal{D}' = \mathcal{D} \setminus (&\{V(p,0) \mid p \in P \cap I\} \cup\\ &\{V(p,1) \mid p \in P \setminus I\} \cup \{S(1), \ldots, S(m)\})\end{aligned}$$

It is immediate to verify that $\mathcal{D}'$ satisfies Σ and that $\mathcal{D}' \cup \{S(m)\}$ is not recoverable from $\langle \Sigma, \mathcal{D} \rangle$. This is proved by the fact that the addition of $S(m)$ creates a sequence of instantiations of the bodies of the first three dependencies of Σ that requires (to keep the consistency of the set) to add to $\mathcal{D}' \cup \{S(m)\}$ first the fact $S(m-1)$, then $S(m-2)$, and so on until $S(1)$: this in turn would imply to also add $S(0)$, but $S(0)$ does not belong to $\mathcal{D}$, which proves that the set $\mathcal{D}' \cup \{S(m)\}$ is

not recoverable from $\langle \Sigma, \mathcal{D} \rangle$. Consequently, there exists a repair of $\langle \Sigma, \mathcal{D} \rangle$ that does not contain $S(m)$.

On the other hand, given a guess of the atoms of the V predicate satisfying the fourth dependency and corresponding to an interpretation of the propositional variables that does not satisfy ϕ, it is immediate to verify that the sequence of instantiations of the bodies of the first three dependencies of Σ mentioned above (which has previously lead to the need of adding $S(0)$ to the set) is blocked by the absence of some fact for V. More precisely: there exists a positive integer $k \leq m$ such that the atoms $S(k), S(k+1), \ldots, S(m)$ can be added to all the repairs corresponding to such a guess of the V atoms. This implies that, if ϕ is unsatisfiable, then $S(m)$ belongs to all the repairs of $\langle \Sigma, \mathcal{D} \rangle$. ∎

Finally, from Theorem 17 above and Theorem 4.1 of [51] (which already holds for inclusion dependencies[5]) it follows that:

Corollary 9. *There exist a set of linear+sticky dependencies and a fact for which* Instance-Checking *is* PTime*-hard in data complexity.*

Upper bounds

Most of the algorithms that we present for IAR-entailment rely on checking if a set of facts is recoverable.

For the general case we present the Algorithm 5 that, given a database $\mathcal{D}$, a set of arbitrary dependencies Σ, and a BUCQ$^{\neq}$ q, checks whether q is IAR-entailed by $\langle \Sigma, \mathcal{D} \rangle$.

Algorithm 5 *IAR-CQEnt*

input: A set of dependencies Σ, a database $\mathcal{D}$, a BUCQ$^{\neq}$ q;
output: A Boolean value;
1: **let** $M_1, \ldots, M_m$ be the images of q in $\mathcal{D}$;
2: **if** there exist m subsets $\mathcal{D}_1, \ldots, \mathcal{D}_m$ of $\mathcal{D}$ such that
every $\mathcal{D}_i$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$
every $M_i \not\subseteq \mathcal{D}_i$ **then**
3: **return** *false*;
4: **else**
5: **return** *true*;

From the definition of IAR semantics, it is easy to prove the correctness of this algorithm.

Proposition 33. *The algorithm IAR-CQEnt*$(\Sigma, \mathcal{D}, q)$ *returns true iff the BUCQ*$^{\neq}$ q *is IAR-entailed by* $\langle \Sigma, \mathcal{D} \rangle$.

Proof. The above algorithm checks whether, for every image M_i of q in $\mathcal{D}$, there exists a repair $\mathcal{D}_i$ such that $M_i \not\subseteq \mathcal{D}_i$. If this is true, every such M_i is not contained in $\mathsf{intRep}_\Sigma(\mathcal{D})$, thus q does not evaluate to true in such an intersection, i.e.

[5] Inclusion dependencies are (equivalent to) single-head linear dependencies without inequalities in which no variable in the body can occur multiple times, thus they are also sticky.

$\langle \Sigma, \mathcal{D} \rangle \not\models^c_{\text{IAR}} q$. Otherwise, the query q has at least one image in $\mathsf{intRep}_\Sigma(\mathcal{D})$, and hence $\langle \Sigma, \mathcal{D} \rangle \models^c_{\text{IAR}} q$. ∎

The above property implies the following upper bounds.

Theorem 26. *IAR-ENT is:*

(*i*) *in* Π^p_2 *in data complexity in the general case of arbitrary dependencies;*

(*ii*) *in coNP in data complexity in the case of full dependencies;*

(*iii*) *in coNP in data complexity in the case of acyclic dependencies.*

Proof. Consider algorithm *IAR-CQEnt*, and recall that m is bounded by n^k, where n is the size of $\mathcal{D}$ and k is the number of predicate atoms of q. Then, theses (*i*), (*ii*) and (*ii*) follow from Proposition 33, other than, respectively, Proposition 4 of [2], Theorem 20 and Corollary 7. ∎

As for linear dependencies, we show the following property (which, by Proposition 28, also holds for AR-entailment).

Theorem 27. *IAR-ENT is in PTIME in data complexity in the case of linear dependencies.*

Proof. The proof follows immediately from Algorithm 4: once computed (in PTIME) the unique repair $\mathcal{D}'$ of $\langle \Sigma, \mathcal{D} \rangle$, the query is then evaluated over $\mathcal{D}'$ (which can be done in AC^0). ∎

Then, we focus on the subclass of acyclic+linear dependencies, and we first show the following property.

Lemma 18. *Let* Σ *be a set of linear dependencies, let* $\mathcal{D}$ *be a database, and let* q *be a BUCQ*$^\neq$*. Then,* $\langle \Sigma, \mathcal{D} \rangle \models^c_{IAR} q$ *(or, equivalently,* $\langle \Sigma, \mathcal{D} \rangle \models^c_{AR} q$*) iff there exists an image* M *of* q *in* $\mathcal{D}$ *such that, for each* $\alpha \in M$*,* $\{\alpha\}$ *is recoverable from* $\langle \Sigma, \mathcal{D} \rangle$*.*

Proof. If the dependencies of Σ are linear, then the repair of $\langle \Sigma, \mathcal{D} \rangle$ is unique. In this case, we have that (*i*) such image M must be recoverable from $\langle \Sigma, \mathcal{D} \rangle$ and (*ii*) a set of facts is recoverable from $\langle \Sigma, \mathcal{D} \rangle$ iff every fact contained in it is recoverable from $\langle \Sigma, \mathcal{D} \rangle$. ∎

Based on the above property, we can use the formula $\Psi^{rec}(\alpha, \Sigma)$ introduced in Section 8.3 to define an FO sentence for deciding query entailment for acyclic+linear dependencies.

Definition 39. [IAR-entailment sentence 1] Given a set of acyclic+linear dependencies Σ and a BUCQ$^\neq$ q, we define the following FO sentence:

$$\Psi^{iar}(q, \Sigma) = \bigvee_{q' \in CQ(q)} \exists \mathbf{x} \left(\gamma \wedge \bigwedge_{\alpha \in predAt(\gamma)} \Psi^{rec}(\alpha, \Sigma) \right)$$

where every BCQ$^\neq$ $q' \in CQ(q)$ is of the form $\exists \mathbf{x}\,(\gamma)$.

The following theorem is an immediate consequence of the above definition of $\Psi^{iar}(q, \Sigma)$, Lemma 16 and Lemma 18.

Theorem 28. *Let Σ be a set of acyclic+linear dependencies, let $\mathcal{D}$ be a database, and let q be a BUCQ$^{\neq}$. Then, $\langle \Sigma, \mathcal{D}\rangle \models^{c}_{IAR} q$ (or, equivalently, $\langle \Sigma, \mathcal{D}\rangle \models^{c}_{AR} q$) iff $\Psi^{iar}(q, \Sigma)$ evaluates to true in $\mathcal{D}$.*

Corollary 10. *IAR-Ent and AR-Ent are in AC^0 in data complexity in the case of acyclic+linear dependencies.*

Example 43. Recall the database $\mathcal{D}$ and the acyclic+linear set Σ of dependencies of Example 41. Note that $\mathcal{D}$ is inconsistent with Σ. Since we are under the conditions of Proposition 28, there is only one minimal way of solving such an inconsistency. Specifically, we must delete the fact $P(\mathsf{p}_2, \mathsf{u}_2)$ and, consequently, $L(\mathsf{u}_1, \mathsf{p}_2)$. Thus, we have that $\mathsf{rep}_\Sigma(\mathcal{D}) = \{\mathsf{intRep}_\Sigma(\mathcal{D})\}$, where $\mathsf{intRep}_\Sigma(\mathcal{D}) = \{U(\mathsf{u}_1, \mathsf{yoda}, \mathsf{d}_1), P(\mathsf{p}_1, \mathsf{u}_1), L(\mathsf{u}_1, \mathsf{p}_1)\}$.

Now, let us take queries $q_1 = \exists p\, P(p, \mathsf{u}_1)$ and $q_2 = \exists u, p, a\, (L(u, p) \wedge P(p, a) \wedge u \neq a)$ and their respective FO rewritings, namely:

$$\begin{aligned}\Psi^{iar}(q_1, \Sigma) &= \exists p\, (P(p, \mathsf{u}_1) \wedge \exists n, r\, U(\mathsf{u}_1, n, r))\\ \Psi^{iar}(q_2, \Sigma) &= \exists u, p\, (L(u, p) \wedge P(p, a) \wedge u \neq a \wedge \exists n, r\, U(a, n, r)\\ &\qquad \exists n, r, a\, (U(u, n, r) \wedge P(p, a) \wedge \exists n', r'\, U(a, n', r'))).\end{aligned}$$

One can verify that q_1 evaluates to true in $\mathsf{intRep}_\Sigma(\mathcal{D})$ as well as $\Psi^{iar}(q_1, \Sigma)$ does in $\mathcal{D}$, and q_2 evaluates to false in $\mathsf{intRep}_\Sigma(\mathcal{D})$ as well as $\Psi^{iar}(q_2, \Sigma)$ does in $\mathcal{D}$. ▽

Finally, for the cases of acyclic+full, full+sticky and full+linear dependencies, we prove a general property that holds for every set of full dependencies Σ that enjoys the CQ-FO-rewritability property.

Let $\mathcal{P} = \{p_1, \ldots, p_m\}$ be a set of predicates, and let k be a positive integer. We define the set of atoms:

$$\mathit{Atoms}(\mathcal{P}, k) = \{p(\mathbf{x}_i) \mid p \in \mathcal{P} \text{ and } i \in \{1, \ldots, k\}\},$$

where each $\mathbf{x}_i$ is a sequence of h fresh variables, if h is the arity of p.

Then, given a CQ-FO-rewritable set of dependencies Σ, a set of predicate atoms $\mathcal{A} \subseteq \mathit{Atoms}(\mathcal{P}, k)$ and a BCQ$^{\neq}$ q of the form $\exists \mathbf{x}\, (\gamma)$ (we assume w.l.o.g. that $\mathbf{x} \cap \mathit{vars}(\mathcal{A}) = \emptyset$), we define the FO formula $\Phi^{ni}(\mathcal{A}, q, \Sigma)$ as follows:

$$\Phi^{ni}(\mathcal{A}, q, \Sigma) = \exists \mathbf{y}\, \Big(\Big(\bigwedge_{\alpha \in \mathcal{A}} \alpha\Big) \wedge \Phi^{rec}(\mathcal{A}, \Sigma) \wedge \neg \Phi^{rec}(\mathcal{A} \cup \mathit{predAt}(q), \Sigma)\Big),$$

where $\mathbf{y}$ is a tuple containing all the variables occurring in $\mathcal{A}$. Note that the variables of $\mathbf{x}$ are free. In words, by existentially closing the above formula and then evaluating it over $\mathcal{D}$ it is possible to check whether there exists a common instantiation for all the atoms of $\mathcal{A}$ such that, under the same assignment, (i) such set of atoms is recoverable and (ii) it is no longer recoverable when we add the atoms of the query. Intuitively, if this happens for some $\mathcal{A}$, then $\langle \Sigma, \mathcal{D}\rangle \not\models^{c}_{\mathrm{IAR}} q$.

Definition 40. [IAR-entailment sentence 2] Let Σ be a CQ-FO-rewritable set of dependencies, let q be a BCQ$^{\neq}$ of the form $\exists \mathbf{x}\, (\gamma)$, and let k be a positive integer. Then $\Phi^{iar}(q, \Sigma, k)$ is the sentence:

$$\Phi^{iar}(q, \Sigma, k) = \exists \mathbf{x}\, \Big(\gamma \wedge \neg\Big(\bigwedge_{\mathcal{A} \subseteq \mathit{Atoms}(\mathit{pred}(\Sigma), k)} \Phi^{ni}(\mathcal{A}, q, \Sigma)\Big)\Big).$$

It is now possible to prove that there exists a k such that, for every database $\mathcal{D}$, $\Phi^{iar}(q, \Sigma, k)$ evaluates to true in $\mathcal{D}$ iff $\langle \Sigma, \mathcal{D} \rangle \models^c_{\text{IAR}} q$. Moreover, the above sentence and property can be immediately extended to the case of BUCQ$^{\neq}$s.

Theorem 29. *IAR-Ent is in AC^0 in data complexity in the case of CQ-FO-rewritable sets of full dependencies.*

Proof. First, it is easy to verify that the following property holds.

Lemma 19. *Let Σ be a CQ-FO-rewritable set of full dependencies, let $\mathcal{A}$ be a set of predicate atoms, let q be a BCQ$^{\neq}$, let $\mathcal{D}$ be a database, and let σ be an instantiation of q in $\mathcal{D}$. Then, $\sigma(\Phi^{ni}(\mathcal{A}, q, \Sigma))$ evaluates to true in $\mathcal{D}$ iff there exists an instantiation σ' of $\bigwedge_{\alpha \in \mathcal{A}} \alpha$ in $\mathcal{D}$ such that $Chase(\sigma'(\mathcal{A}), \Sigma) \subseteq \mathcal{D}$ and $Chase(\sigma'(\mathcal{A}) \cup \sigma(predAt(q)), \Sigma) \not\subseteq \mathcal{D}$.*

Then, the following property follows from the previous lemma and from the fact that, for every database $\mathcal{D}'$ of size not greater than k, there exists a subset $\mathcal{A}$ of $Atoms(pred(\Sigma), k)$ such that $\mathcal{D}'$ is an image of $\mathcal{A}$.

Lemma 20. *Let Σ be a CQ-FO-rewritable set of full dependencies, let q be a BCQ$^{\neq}$, let k be a positive integer, and let $\mathcal{D}$ be a database. Then, $\Phi^{iar}(q, \Sigma, k)$ evaluates to true in $\mathcal{D}$ iff there exists an image M of q in $\mathcal{D}$ such that, for every database $\mathcal{D}'$ such that $\mathcal{D}' \subseteq \mathcal{D}$ and $|\mathcal{D}'| \leq k$, if $Chase(\mathcal{D}', \Sigma) \subseteq \mathcal{D}$ then $Chase(\mathcal{D}' \cup M, \Sigma) \subseteq \mathcal{D}$.*

Finally, if Σ is CQ-FO-rewritable, then for every database $\mathcal{D}$ and for every BCQ$^{\neq}$ q, if there exists a subset $\mathcal{D}'$ of $\mathcal{D}$ such that $Chase(\mathcal{D}', \Sigma) \subseteq \mathcal{D}$ and $Chase(\mathcal{D}' \cup M, \Sigma) \not\subseteq \mathcal{D}$, then there exists a subset $\mathcal{D}''$ of $\mathcal{D}$ such that $Chase(\mathcal{D}'', \Sigma) \subseteq \mathcal{D}$ and $Chase(\mathcal{D}'' \cup M, \Sigma) \not\subseteq \mathcal{D}$ and $|\mathcal{D}''| \leq k$, where k is an integer that depends on Σ and q (so it does not depend on $\mathcal{D}$).

This property and Lemma 20 imply that BCQ$^{\neq}$ entailment under IAR semantics is FO-rewritable, and thus in AC^0 in data complexity.

The FO-rewritability property straightforwardly extends to the case when the query is a BUCQ$^{\neq}$ q, as $\langle \Sigma, \mathcal{D} \rangle \models^c_{\text{IAR}} q$ iff there exists a BCQ$^{\neq}$ $q' \in CQ(q)$ such that $\langle \Sigma, \mathcal{D} \rangle \models^c_{\text{IAR}} q'$. ∎

As a consequence of the above theorem and of Proposition 29, we get the following property.

Corollary 11. *IAR-Ent is in AC^0 in data complexity in the case of either acyclic+full, or full+linear or full+sticky dependencies.*

8.6 AR-entailment

We now turn our attention to the problem of AR-entailment of BUCQ$^{\neq}$s.

As for the lower bounds for AR-entailment, almost all of them follow from the ones already shown for the instance checking problem. Here, we show the following property.

Theorem 30. *There exist a set of acyclic+full+guarded+sticky dependencies and a BUCQ for which AR-Ent is coNP-hard in data complexity.*

Proof. The following reduction from 3-SAT uses a set of dependencies Σ that is acyclic+full+guarded+sticky:

$$\begin{gathered}\forall x\ (V(x,0) \wedge V(x,1) \rightarrow U(x)) \\ \forall z, x_1, v_1, x_2, v_2, x_3, v_3\ (NC(z, x_1, v_1, x_2, v_2, x_3, v_3) \rightarrow V(x_1, v_1)) \\ \forall z, x_1, v_1, x_2, v_2, x_3, v_3\ (NC(z, x_1, v_1, x_2, v_2, x_3, v_3) \rightarrow V(x_2, v_2)) \\ \forall z, x_1, v_1, x_2, v_2, x_3, v_3\ (NC(z, x_1, v_1, x_2, v_2, x_3, v_3) \rightarrow V(x_3, v_3))\end{gathered}$$

Then, given a 3-CNF formula ϕ with m clauses, in the database $\mathcal{D}$ we represent every clause of ϕ with a fact NC: e.g. if the i-th clause of ϕ is $\neg a \vee b \vee \neg c$, we add the fact $NC(i, a, 1, b, 0, c, 1)$. Moreover, the database contains the facts $V(a,0), V(a,1)$ for every propositional variable $a \in \mathit{vars}(\phi)$.

Now, given the following BCQ:

$$q = \exists z, x_1, v_1, x_2, v_2, x_3, v_3\ NC(z, x_1, v_1, x_2, v_2, x_3, v_3)$$

it is easy to verify that ϕ is unsatisfiable iff $\langle \Sigma, \mathcal{D} \rangle \models^{c}_{\mathrm{AR}} q$. ∎

Notice that, as a consequence of the previous theorem, both in the case of acyclic+full dependencies and in the case of full+sticky dependencies, AR-entailment of BUCQ$^{\neq}$s is coNP-hard, and therefore computationally harder than IAR-entailment of BUCQ$^{\neq}$s (which is in AC^0 in both cases).

Moreover, we establish the following upper bounds for AR-entailment in the case of arbitrary, full, and acyclic dependencies.

Theorem 31. *AR-Ent is*

(*a*) *in* Π^p_2 *in data complexity in the general case of arbitrary dependencies;*

(*b*) *in coNP in data complexity in the case of either full or acyclic dependencies.*

Proof. Let q be a BUCQ$^{\neq}$. From the definition of AR-entailment, it follows that $\langle \Sigma, \mathcal{D} \rangle \not\models^{c}_{\mathrm{AR}} q$ iff there exists a subset $\mathcal{D}'$ of $\mathcal{D}$ such that (*i*) $\mathcal{D}'$ is a repair of $\langle \Sigma, \mathcal{D} \rangle$ and (*ii*) q evaluates to false in $\mathcal{D}'$. Therefore, the theses (*a*) and (*b*) follow from the fact that repair checking is, respectively, in coNP and in PTime, and that evaluating a BUCQ$^{\neq}$ can be done in AC^0. ∎

The complexity results established so far are summarized by Table 8.1. Such findings allow us to draw a complete picture of the data complexity of repair checking and both AR-entailment and IAR-entailment of BUCQ$^{\neq}$s, for all the possible combinations of the five classes of existential rules considered in this chapter.

We remark that, with the exception of the cases in AC^0, every outcome is a completeness result for the complexity class reported. Every such result is actually new. However, some of them (the ones marked with numbers) extend previously known lower and/or upper bounds. More precisely:

(1) Extends the lower bound proved in [51] for denials;

(2) Extends the upper bound proved in [114] for non-disjunctive LAV TGDs without inequalities;

(3) Extends the lower bound proved in [2] for full TGDs and EGDs;

Existential rules class	RC	IC	$\text{BUCQ}^{\neq}$ $\models^{c}_{\text{IAR}}$	$\text{BUCQ}^{\neq}$ $\models^{c}_{\text{AR}}$
A+L[+S]	in AC^0	in AC^0	in AC^0	
F+L[+S]	in AC^0	in AC^0	in AC^0	
A+F[+G][+S]	in AC^0	in AC^0	in AC^0	coNP (1)
F+S[+A][+G]	in AC^0	in AC^0	in AC^0	coNP (1)(5)
L[+S]	PTime (2)	PTime	PTime (2)	
A[+G][+S]	in AC^0	coNP	coNP	coNP (1)
F[+G]	PTime (3)(4)	coNP	coNP	coNP (1)(5)
All[+G][+S]	coNP (1)(6)	Π^p_2 (1)	Π^p_2 (1)	Π^p_2 (1)(2)

A = Acyclic, F = Full, G = Guarded, L = Linear, S = Sticky

Table 8.1. Data complexity results for all the classes of existential rules considered in this work. Joined cells indicate that the two entailment problems coincide (see Proposition 28). All entries are completeness results, except for the AC^0 ones. And the [+X] notation indicates that the same complexity holds if the considered class of dependencies is also a subclass of X. We recall that L implies G (i.e. linear dependencies are a subclass of guarded dependencies).

(4) Extends the upper bound proved in [110] for denials;

(5) Extends the upper bound proved in [114] for GAV TGDs;

(6) The upper bound has been established in [2].

Chapter 9

Consistent query answering with open and closed predicates

In this chapter, we explore the possibility of integrating open- and closed-world assumptions by adopting both open and closed predicates in the same knowledge base. This idea has already been considered in several works[1] but not, as far as we know, in the context of CQA.

We identify a syntactic condition for dependencies for which we are able to integrate our previous findings with the ones provided in [88] for the purely-open case. We provide data complexity results for both the standard query entailment problem (under this "mixed" semantics) and for the analogous of AR and IAR entailment problems.

9.1 CQA semantics with open and closed predicates

Besides the case in which pure closed-world assumption is adopted, we are interested on how integrating open and closed-world semantics. As discussed in Section 10, this is typically achieved by partitioning the signature into open and closed predicates [95].

Hereafter, we assume that all the databases, all the sets of dependencies and all the FO sentences are given for a *mixed signature* $\mathcal{S}_o \cup \mathcal{S}_c$, where $\mathcal{S}_o$ is the set of *open predicates* and $\mathcal{S}_c$ is the set of *closed predicates*, such that $\mathcal{S}_o \cap \mathcal{S}_c = \emptyset$ and $\mathcal{S}_c$ always includes $\bot$. Moreover, given a database $\mathcal{D}$, we indicate as $\mathcal{D}_c$ the projection of $\mathcal{D}$ over $\mathcal{S}_c$.

Definition 41. [OC-model] Given a set of dependencies Σ and two databases $\mathcal{D}$ and $\mathcal{D}'$, we say that $\mathcal{D}'$ is a *OC-model* of $\langle \Sigma, \mathcal{D} \rangle$ if:

(i) the Herbrand interpretation of $\mathcal{D}'$ is a model of the FO theory $\Sigma \cup \mathcal{D}$;

(ii) $\mathcal{D}'_c = \mathcal{D}_c$.

Moreover, we say that $\langle \Sigma, \mathcal{D} \rangle$ is OC-consistent if it has at least one OC-model.

[1] We refer the reader to Section 10.2 for a detailed review of the related work.

As for query entailment, we adopt the usual skeptical semantics, which reduces to classical (open) entailment when $\mathcal{S}_c$ is empty or it just contains $\bot$. Specifically, we study entailment BUCQs *without* inequalities.

Definition 42. [OC-entailment] Given a set of dependencies Σ and a BUCQ q, we write $\langle \Sigma, \mathcal{D} \rangle \models^{oc} q$ if q evaluates to true over all the OC-models of $\langle \Sigma, \mathcal{D} \rangle$.

We now introduce the natural extension of Definition 32 to the mixed semantics.

Definition 43. [OC-repair] Given a database $\mathcal{D}$ and a set of dependencies Σ, an *OC-repair* of $\langle \Sigma, \mathcal{D} \rangle$ is a maximal subset $\mathcal{D}'$ of $\mathcal{D}$ such that $\langle \Sigma, \mathcal{D}' \rangle$ is OC-consistent.

We also extend the functions $\mathsf{rep}_\Sigma(\cdot)$ and $\mathsf{intRep}_\Sigma(\cdot)$ in order to work with predicates for a mixed signature. More precisely, we denote by $\mathsf{rep}^{oc}_\Sigma(\mathcal{D})$ the set of all the possible OC-repairs of $\langle \Sigma, \mathcal{D} \rangle$. It is again immediate to verify that an OC-repair always exists for every $\langle \Sigma, \mathcal{D} \rangle$ (since the empty database is an OC-model for every possible set of dependencies). Moreover, we will denote as $\mathsf{intRep}^{oc}_\Sigma(\mathcal{D})$ the set $\bigcap_{\mathcal{D}' \in \mathsf{rep}^{oc}_\Sigma(\mathcal{D})} \mathcal{D}'$.

We are now ready to re-define the AR and IAR entailment semantics.

Definition 44. [OC-AR and OC-IAR-entailment] Given as input a database $\mathcal{D}$, a set Σ of dependencies, and a BUCQ q:

(*i*) OC-AR-Ent is the problem of deciding whether $\langle \Sigma, \mathcal{D}' \rangle \models^{oc} q$ for every $\mathcal{D}' \in \mathsf{rep}^{oc}_\Sigma(\mathcal{D})$. In case this holds, we write $\langle \Sigma, \mathcal{D} \rangle \models^{oc}_{\text{AR}} q$;

(*ii*) OC-IAR-Ent is the problem of deciding whether $\langle \Sigma, \mathsf{intRep}^{oc}_\Sigma(\mathcal{D}) \rangle \models^{oc} q$. In case this holds, we write $\langle \Sigma, \mathcal{D} \rangle \models^{oc}_{\text{IAR}} q$.

Example 44. Consider the set of constraints and the set of facts (let us denote them with Σ and $\mathcal{D}$, respectively) of Example 3. As explained in the aforementioned example, we have two possible repairs, namely:

$$\mathsf{rep}^{oc}_\Sigma(D) = \{\{G(\mathsf{ann}), G(\mathsf{sam}), E(\mathsf{tom}, \mathsf{nlp}, 2023), E(\mathsf{tom}, \mathsf{kr}, 2023)\}, \\ \{G(\mathsf{ann}), R(\mathsf{sam}), E(\mathsf{tom}, \mathsf{nlp}, 2023), E(\mathsf{tom}, \mathsf{kr}, 2023)\}\}.$$

Then, one can see that the BUCQ $G(\mathsf{sam}) \vee R(\mathsf{sam})$ is OC-AR-entailed but not OC-IAR-entailed by $\langle \Sigma, \mathcal{D} \rangle$. Moreover, the presence of $G(\mathsf{ann})$ in $\mathsf{intRep}^{oc}_\Sigma(\mathcal{D})$ causes the query $\exists x\,(F(x) \wedge L(x))$ to be OC-IAR-entailed by $\langle \Sigma, \mathcal{D} \rangle$, despite no facts with predicate F or L occur in $\mathcal{D}$. ▽

From now on, we denote the AR- and IAR-entailment when all predicates are interpreted as open (i.e. the semantics studied in [88]) with the symbols $\models^{o}_{\text{AR}}$ and $\models^{o}_{\text{IAR}}$, respectively.

The following property immediately follows from the previous definitions.

Proposition 34. *Let Σ be a set of dependencies, and let $\mathcal{D}$ be a database. If $pred(\Sigma \cup \mathcal{D}) \cap \mathcal{S}_o = \emptyset$, then $\langle \Sigma, \mathcal{D} \rangle \models^{oc}_{AR} q$ iff $\langle \Sigma, \mathcal{D} \rangle \models^{c}_{AR} q$ and $\langle \Sigma, \mathcal{D} \rangle \models^{oc}_{IAR} q$ iff $\langle \Sigma, \mathcal{D} \rangle \models^{c}_{IAR} q$. Moreover, if $pred(\Sigma \cup \mathcal{D}) \cap \mathcal{S}_c = \emptyset$, then $\langle \Sigma, \mathcal{D} \rangle \models^{oc}_{AR} q$ iff $\langle \Sigma, \mathcal{D} \rangle \models^{o}_{AR} q$ and $\langle \Sigma, \mathcal{D} \rangle \models^{oc}_{IAR} q$ iff $\langle \Sigma, \mathcal{D} \rangle \models^{o}_{IAR} q$.*

Finally, we remark the following important property of the chase with open dependencies.

Proposition 35. *Let Σ be a set of TGDs and let $\mathcal{D}$ be a database. Then, for every BUCQ q, $\Sigma_o \cup \mathcal{D} \models q$ iff q evaluates to true in $Chase(\mathcal{D}, \Sigma)$.*

9.2 Open-closed-separable dependencies

We say that a dependency τ is:

- a *closed-head* dependency if no open predicate occurs in $\mathit{head}(\tau)$;
- an *open-head* dependency if all the atoms occurring in $\mathit{head}(\tau)$ are atoms with an open predicate.

Given a set Σ of dependencies, we denote by Σ_c (resp., Σ_o) the set of closed-head (resp., open-head) dependencies occurring in Σ.

In the rest of this section, we consider sets of dependencies Σ where every $\tau \in \Sigma$ is such that, if an open predicate occurs in the head of τ, then τ is non-disjunctive and no inequality occurs in it.[2] Under this assumption, every set of dependencies Σ can be equivalently (w.r.t. the entailment task that we study) transformed into a set of dependencies Σ' such that $\Sigma' = \Sigma'_o \cup \Sigma'_c$. In other words, all the dependencies having a "mixed" head can be split into dependencies without mixed heads. For example, given a dependency $\tau = \forall \mathbf{x}(\gamma(\mathbf{x}) \rightarrow \exists y, z, w\,(P_o(y, z) \wedge P_c(z, w)))$, where $\gamma(\mathbf{x})$ is any conjunction of atoms, $P_o \in \mathcal{S}_o$ and $P_c \in \mathcal{S}_c$, then τ is equivalent to the following set of dependencies:

$$\begin{aligned} \forall \mathbf{x}\,(\gamma(\mathbf{x}) \rightarrow \exists y, z, w\, O^{aux}(y, z, w)) \\ \forall y, z, w\,(O^{aux}(y, z, w)) \rightarrow P_o(y, z) \\ \forall y, z, w\,(O^{aux}(y, z, w)) \rightarrow P_c(z, w) \end{aligned}$$

where O^{aux} is a fresh open predicate.

As usual, in the following we call *frontier variables* of a dependency τ all the variables occurring both in the body and in the head of τ. Moreover, we define the set of *existential positions* of a set of dependencies Σ as the minimal set $\mathcal{P}$ of positions of open predicates occurring in Σ such that:

- if there exists a dependency $\tau \in \Sigma_o$ such that there is an atom with predicate p in $\mathit{head}(\tau)$ whose i-th argument is an existential head variable, then $p[i] \in \mathcal{P}$;
- if there exists a dependency $\tau \in \Sigma_o$ such that there is a frontier variable x of τ only occurring in existential positions in $\mathit{body}(\tau)$ and occurring in the i-th position of an atom with predicate p in $\mathit{head}(\tau)$, then $p[i] \in \mathcal{P}$.

Every position of predicates occurring in Σ that is not an existential position of Σ is called a *non-existential position* of Σ. Furthermore, we say that a variable occurring in a dependency $\tau \in \Sigma$ is *grounded* in (τ, Σ) if it occurs in $\mathit{body}(\tau)$ in at least one non-existential position of Σ.

Definition 45. [OCS] Given a set Σ of dependencies, we say that Σ is *open-closed-separable* (*OCS*) if, for every $\tau \in \Sigma_c$, all the frontier variables of τ and all the variables occurring in inequality atoms in $\mathit{body}(\tau)$ are grounded in (τ, Σ).

[2] This assumption is necessary in order to make our framework coincide with the one studied in [88] in the case when there are no closed predicates.

It is easy to verify that a non-existential position is a position in which, for every database $\mathcal{D}$, no labeled null can occur in $\mathit{Chase}(\mathcal{D}, \Sigma_o)$. Informally, this property implies a "controlled" interaction between Σ_c and Σ_o when Σ is OCS: in particular, the dependencies in Σ_c have effect on the OC-consistency of $\langle \Sigma, \mathcal{D} \rangle$, but they have no effect on OC-entailment in the case when $\langle \Sigma, \mathcal{D} \rangle$ is OC-consistent. Moreover, when Σ is OCS, checking the consistency of $\langle \Sigma, \mathcal{D} \rangle$ can be done by "separating" reasoning over Σ_c and Σ_o (see Lemma 22 below).

Example 45. Let us consider again the setting illustrated in Example 3. It is straightforward to verify that the described set of dependencies (let us indicate it with Σ) is OCS. In particular, the only existential position in Σ is the second argument of the open predicate F, and in the second dependency (let us call it τ_2) the variable c occurring in such a position is not grounded in (τ_2, Σ). Nevertheless, c is not a frontier variable of τ_2 and it does not participate in any inequality of the body of τ_2. ▽

In the remaining part of this chapter, we study the data complexity of the problems introduced in Section 9.1 in case dependencies are OCS.

9.2.1 OC-entailment

We now study the complexity of OC-entailment under OCS dependencies. To this aim, we show the following property for such a class of dependencies.

Lemma 21. *Let Σ be a set of OCS dependencies, let $\mathcal{D}$ be a database such that $\langle \Sigma, \mathcal{D} \rangle$ is OC-consistent, and let q be a BUCQ. Then, $\langle \Sigma, \mathcal{D} \rangle \models^{oc} q$ iff $\Sigma \cup \mathcal{D} \models q$ iff $\Sigma_o \cup \mathcal{D} \models q$.*

Proof. By definition, $\langle \Sigma, \mathcal{D} \rangle \models^{oc} q$ iff q evaluates to true in all the OC-models of $\langle \Sigma, \mathcal{D} \rangle$. Anyway, every such OC-model $\mathcal{D}'$ (at least one of which must exist, by the OC-consistency hypothesis) is such that $\mathcal{D}'_c = \mathcal{D}_c$, which implies that it can be computed by only chasing $\mathcal{D}$ w.r.t. Σ_o and replace with suitable constants all the existential variables occurring in the heads of the applied rules. Therefore, we have that $\mathit{Chase}(\mathcal{D}, \Sigma_o)$ produces a canonical model for the OC-models of $\langle \Sigma, \mathcal{D} \rangle$. Thus, $\langle \Sigma, \mathcal{D} \rangle \models^{oc} q$ iff q evaluates to true in $\mathit{Chase}(\mathcal{D}, \Sigma_o)$, which, by Proposition 35, is true iff $\Sigma_o \cup \mathcal{D} \models q$.

Now note that, for every set of facts $\mathcal{D}'$ whose Herbrand interpretation is a model of the FO theory $\Sigma \cup \mathcal{D}$, there exists an OC-model $\mathcal{D}''$ of $\langle \Sigma, \mathcal{D} \rangle$ such that $\mathcal{D}' \supseteq \mathcal{D}''$. For obtaining such $\mathcal{D}''$ it is indeed sufficient to remove from $\mathcal{D}'$ all the facts over $\mathcal{S}_c$ that do not belong to $\mathcal{D}_c$. Therefore, if $\langle \Sigma, \mathcal{D} \rangle \models^{oc} q$, i.e. q evaluates to true in each OC-model of $\langle \Sigma, \mathcal{D} \rangle$, then a fortiori (by monotonicity of BUCQ evaluation) it evaluates to true in each such $\mathcal{D}'$ (i.e. $\Sigma \cup \mathcal{D} \models q$). On the other direction, we have that $\Sigma \cup \mathcal{D} \models q$ clearly implies $\langle \Sigma, \mathcal{D} \rangle \models^{oc} q$, which proves the thesis. ∎

Given an OCS set of dependencies Σ and a dependency $\tau \in \Sigma$, let $\mathit{body}_o(\tau)$ denote the CQ corresponding to the conjunction of the atoms with open predicates of $\mathit{body}(\tau)$ having as existentially quantified variables all the variables that are not grounded in (τ, Σ). Similarly, $\mathit{body}_c(\tau)$ denotes the (not necessarily safe) $\mathrm{CQ}^{\neq}$ corresponding to the conjunction of the atoms with closed predicates or inequalities of

$body(\tau)$. Also, let $grBody(\Sigma, \mathcal{D})$ denote the dependencies obtained from Σ_c instantiating, for every $\tau \in \Sigma_c$, all the variables grounded in (τ, Σ) with constants of $\mathcal{D}$ in all the possible ways. Notice that, if $\tau \in grBody(\Sigma, \mathcal{D})$, then $body_o(\tau)$ is a BCQ, $body_c(\tau)$ is a ground BCQ$^{\neq}$ and $head(\tau)$ is a BCQ$^{\neq}$.

Lemma 22. *Let Σ be a set of OCS dependencies. Then, $\langle \Sigma, \mathcal{D} \rangle$ is OC-consistent iff, for each $\tau \in grBody(\Sigma_c, \mathcal{D})$, if $\langle \Sigma_o, \mathcal{D} \rangle \models body_o(\tau)$ and $body_c(\tau)$ evaluates to true in $\mathcal{D}$, then $head(\tau)$ evaluates to true in $\mathcal{D}$.*

Proof. Suppose $\langle \Sigma, \mathcal{D} \rangle$ is not OC-consistent. Then, it is immediate to verify that $Chase(\mathcal{D}, \Sigma_o)$ does not satisfy Σ_c, which implies that there exists a dependency $\tau \in \Sigma_c$ and a substitution σ of the variables of τ with the constants and labeled nulls occurring in $Chase(\mathcal{D}, \Sigma_o)$ such that $\sigma(body(\tau))$ evaluates to true in $Chase(\mathcal{D}, \Sigma_o)$ and $\sigma(head(\tau))$ evaluates to false in $Chase(\mathcal{D}, \Sigma_o)$. Now, from Definition 45 it immediately follows that there exists a substitution σ with the above properties and such that all the frontier variables of τ and all the variables occurring in some inequality of $body(\tau)$ are replaced by σ with a constant. Finally, the thesis is implied by the last property and the fact that, for every BCQ q, $\Sigma_o \cup \mathcal{D} \models q$ iff q evaluates to true in $Chase(\mathcal{D}, \Sigma_o)$.

The other direction of the proof is trivial. ∎

From the above two lemmas, we can establish the following complexity results.

Proposition 36. *Let Σ be a set of OCS dependencies. OC-entailment of BUCQs is:*

(*i*) *in AC^0 in data complexity in case the dependencies of Σ_o are either acyclic, or linear, or sticky;*

(*ii*) *PTime-complete in data complexity in case the dependencies of Σ_o are either full or guarded.*

Proof. The known PTime results for data complexity of BCQ entailment under OWA [40] imply both the lower bound and, along with Lemma 22, the upper bound for the cases when Σ_o is either guarded or full are in PTime.

Now, suppose Σ is OCS and Σ_o is either acyclic, or linear, or sticky. For each of these cases it is well-known [40, 41] that conjunctive query answering is first-order rewritable, i.e. there exists a query rewriting algorithm *FO-Rewrite* that, given a BUCQ q and Σ_o, returns an FO sentence ϕ such that, for every database $\mathcal{D}$, $\Sigma_o \cup \mathcal{D} \models q$ iff ϕ evaluates to true in $\mathcal{D}$. This immediately implies that $\langle \Sigma, \mathcal{D} \rangle$ is OC-consistent iff the following sentence $\Psi_{oc}^{cons}(\Sigma)$ evaluates to true in $\mathcal{D}$:

$$\Psi_{oc}^{cons}(\Sigma) = \bigwedge_{\tau \in \Sigma_c} \forall \mathbf{x} \left((\textit{FO-Rewrite}(body_o(\tau)(\mathbf{x}_1), \Sigma_o) \wedge body_c(\tau)(\mathbf{x}_2)) \rightarrow head(\tau)(\mathbf{x}_3) \right),$$

where $\mathbf{x}_1$ are the free variables of $body_o(\tau)$, $\mathbf{x}_2$ are the free variables of $body_c(\tau)$, $\mathbf{x}_3$ are the frontier variables of τ, $\mathbf{x} = \mathbf{x}_1 \cup \mathbf{x}_2$ are the variables grounded in (τ, Σ) (thus $\mathbf{x}_3 \subseteq \mathbf{x}$), and $\textit{FO-Rewrite}(body_o(\tau)(\mathbf{x}_1), \Sigma_o)$ denotes the FO formula returned by the algorithm *FO-Rewrite* executed on the CQ $body_o(\tau)(\mathbf{x}_1)$ with respect to the dependencies Σ_o. This shows, together with Lemma 22, that the cases when Σ_o is either acyclic, or linear, or sticky are in AC^0. ∎

9.2.2 Repair checking

As for the problem analogous to repair checking, denoted as OC-Rep-Checking, we establish a tight complexity result for the case in which no restrictions are applied to dependencies in Σ_c.

Theorem 32. *Let Σ be a set of OCS dependencies. OC-Rep-Checking is coNP-complete in data complexity in case the set of dependencies Σ_o is either acyclic, or full, or guarded, or linear, or sticky.*

Proof. As for the upper bound, consider the nondeterministic procedure that decides whether $\mathcal{D}'$ is an OC-repair of $\langle \Sigma, \mathcal{D} \rangle$ by trivially applying the definition: check whether $\langle \Sigma, \mathcal{D}' \rangle$ is OC-consistent and there exists no $\mathcal{D}''$ such that $\mathcal{D}' \subset \mathcal{D}'' \subseteq \mathcal{D}$ and $\langle \Sigma, \mathcal{D}'' \rangle$ is OC-consistent. Since Lemma 22 implies that, in all the cases under examination, consistency is in PTime in data complexity, the above technique establishes a coNP upper bound for the OC-Rep-Checking problem. The lower bound follows from the known coNP lower bound for Rep-Checking in the case when Σ_o is empty (Theorem 19). ∎

9.2.3 OC-AR-entailment

As for BUCQ answering, we show that OCS dependencies enjoy the following crucial property when Σ_c only consists of denials, which allows us to reduce both OC-AR- and OC-IAR-entailment to the purely-open scenario.

Lemma 23. *Let Σ be an OCS set of dependencies such that Σ_c is a set of denials, and let $\mathcal{D}$ be a database. Then, for every BUCQ q, $\langle \Sigma, \mathcal{D} \rangle \models^{oc}_{AR} q$ iff $\langle \Sigma, \mathcal{D} \rangle \models^{o}_{AR} q$, and $\langle \Sigma, \mathcal{D} \rangle \models^{oc}_{IAR} q$ iff $\langle \Sigma, \mathcal{D} \rangle \models^{o}_{IAR} q$.*

Proof. Again, it is immediate to verify that, for every database $\mathcal{D}'$, $\Sigma \cup \mathcal{D}'$ is a consistent FO theory (under OWA) iff $\mathit{Chase}(\mathcal{D}', \Sigma)$ does not contain $\bot$ iff $\langle \Sigma, \mathcal{D}' \rangle$ is OC-consistent. This implies that the set of repairs of $\langle \Sigma, \mathcal{D}' \rangle$ coincides with its set of OC-repairs, which in turn implies the thesis. ∎

Theorem 33. *Let Σ be an OCS set of dependencies such that Σ_c is a set of denials, and let $\mathcal{D}$ be a database. Then, OC-AR-Ent is coNP-complete in data complexity in the case Σ_o is either acyclic, or full, or guarded, or linear or sticky.*

Proof. The thesis directly derives from Lemma 23 and from the results of [88].[3] ∎

We also prove that, if Σ_o is either acyclic, full, guarded, linear or sticky and none of such restrictions is applied to Σ_c, then the data complexity of OC-AR-entailment of BUCQs does not increase with respect to the purely-closed case.

Theorem 34. *Let Σ be a set of OCS dependencies. Then, OC-AR-Ent is Π^p_2-complete in data complexity in the case Σ_o is either acyclic, or full, or guarded, or linear, or sticky.*

[3] Actually, with respect to the cases studied in [88], our language also allows for the presence of inequalities in the body of the dependencies: however, when Σ is OCS, such inequalities only involve grounded variables, which implies that all the upper bounds shown in [88] can be easily extended to the presence of such inequalities in the bodies of dependencies.

Proof. First, $\langle \Sigma, \mathcal{D} \rangle \not\models^{oc}_{\text{AR}} q$ iff there exists $\mathcal{D}' \subseteq \mathcal{D}$ such that $\mathcal{D}'$ is an OC-repair of $\langle \Sigma, \mathcal{D} \rangle$ and $\Sigma_o \cup \mathcal{D}' \not\models q$. Now, since by Theorem 32 OC-Rep-Checking is in coNP in data complexity, and since $\Sigma_o \cup \mathcal{D}' \not\models q$ can be checked in PTime in data complexity in all the cases considered, the Π^p_2 upper bound in data complexity follows.

The Π^p_2 lower bound follows immediately from the fact that, in the case when Σ_o is empty, then OC-AR-entailment of BUCQs coincides with AR-entailment of BUCQs when all predicates are interpreted as closed, which implies by previously mentioned results (e.g. the lower bound proved in [51] for denials) that the data complexity of OC-AR-Ent (and already for instance checking) is Π^p_2-complete. ∎

9.2.4 OC-IAR-entailment

We now consider OC-IAR-entailment of BUCQs when Σ is an OCS set of dependencies. As anticipated in the previous section, in the case Σ_c only consists of denials, Lemma 23 allows us to extend the results of [88] to the scenario with open and closed predicates. In particular, if the dependencies in Σ_o are either acyclic, linear, or sticky, then OC-IAR-entailment is FO rewritable.

Theorem 35. *Let Σ be an OCS set of dependencies such that Σ_c is a set of denials, and let $\mathcal{D}$ be a database. Then, OC-IAR-Ent is:*

- *in AC^0 in data complexity in the case Σ_o is either acyclic, or linear or sticky;*
- *coNP-complete in data complexity in the case Σ_o is either full or guarded.*

Proof. The proof immediately follows from Lemma 23 and from the complexity results provided in [88]. ∎

Similarly to OC-AR-entailment, if Σ_o is either acyclic, full, guarded, linear or sticky and none of such restrictions is applied to Σ_c, then the data complexity of OC-IAR-entailment of BUCQs does not increase with respect to the purely-closed case.

Theorem 36. *Let Σ be a set of OCS dependencies. OC-IAR-Ent is Π^p_2-complete in data complexity in the case Σ_o is either acyclic, full, guarded, linear or sticky.*

Proof. First, we call image of a BUCQ q in $\langle \Sigma_o, \mathcal{D} \rangle$ a minimal subset M of $\mathcal{D}$ such that $\Sigma_o \cup M \models q$.

Now, consider the general nondeterministic technique described in Algorithm 6.

Since OC-Rep-Checking in all the cases considered by the theorem is coNP-complete in data complexity, and since for all such cases deciding whether $\Sigma_o \cup \mathcal{D} \models q$ is in PTime in data complexity, the above technique provides a Π^p_2 upper bound for the data complexity of OC-IAR-Ent.

Analogously to the proof of Theorem 34, the Π^p_2 lower bound derives from the case when all predicates are interpreted as closed. ∎

The main complexity results for OCS dependencies established by the above properties are summarized in Table 9.1.

Algorithm 6 *IAR-Mixed-Ent*

input: A set of dependencies Σ, a database $\mathcal{D}$, a BUCQ q;
output: A Boolean value;
1: **let** $M_1, \ldots, M_m$ be the images of q in $\langle \Sigma_o, \mathcal{D} \rangle$;
2: **if** there exist $\mathcal{D}'$ subset of $\mathcal{D}$ (where $\mathcal{D} \setminus \mathcal{D}' = \{\alpha_1, \ldots, \alpha_m\}$)
and m subsets $\mathcal{D}_1, \ldots, \mathcal{D}_m$ of $\mathcal{D}$ such that
for every i s.t. $1 \leq i \leq m$, $\mathcal{D}_i$ is an OC-repair of $\langle \Sigma, \mathcal{D} \rangle$ and
for every i s.t. $1 \leq i \leq m$, $\alpha_i \notin \mathcal{D}_i$
and $\Sigma_o \cup \mathcal{D} \not\models q$ **then**
3: **return** $false$;
4: **else**
5: **return** $true$;

Σ_o	Σ_c	BUCQ $\models^{oc}$	BUCQ $\models^{oc}_{\text{IAR}}$	BUCQ $\models^{oc}_{\text{AR}}$
A	D	in AC^0	in AC^0 (7)	coNP (7)
L	D	in AC^0	in AC^0 (7)	coNP (7)
S	D	in AC^0	in AC^0 (7)	coNP (7)
F	D	PTIME	coNP (7)	coNP (7)
G	D	PTIME	coNP (7)	coNP (7)
A	All	in AC^0	Π^p_2	Π^p_2
L	All	in AC^0	Π^p_2	Π^p_2
S	All	in AC^0	Π^p_2	Π^p_2
F	All	PTIME	Π^p_2	Π^p_2
G	All	PTIME	Π^p_2	Π^p_2

A = Acyclic, F = Full, G = Guarded,
L = Linear, S = Sticky, D = Denials

Table 9.1. Data complexity results for OCS sets of dependencies. All entries are completeness results, except for the AC^0 ones.
(7) these results are easily derived from [88].

Chapter 10

Related work

10.1 Controlled query evaluation

The CQE approach has been introduced by Sicherman *et al.* in [108]. In the foundational work, the authors provide a general framework for privacy-preserving query answering and discuss some criteria that can be used by a censor to alter user queries, under various assumptions on the knowledge of the attacker about the information system. Data are interpreted under closed world assumption, censors may refuse to provide an answer to a query but never lie to users, who can have additional background knowledge about the data in the database, and can even know which criterion is used by the censor. The relationship between the lying approach, i.e. when the system may return the negation of correct answers, and the refusal one, i.e. when the system can refuse to answer some queries, has been then investigated for propositional closed databases in [29, 32], for the case in which the attacker does not know the policy, and in [30, 31], for known policies.

Similarly as done in the present thesis, some previous works on CQE have proposed techniques to conceal sensitive data in a manner that makes it impossible for a user to discern the actual database or knowledge base from an alternative version without secrets, as, e.g., in [31, 32, 34, 37, 112]. Using the terminology proposed in this thesis, we may say that all such papers study censors that, even if defined in different, often incomparable ways, enjoy the indistinguishability property that we formulate in Definition 6. As pointed out in [37] and more recently in [35], and as we already remarked in previous sections, censors of this form proved to be more robust than others: censors not enjoying indistinguishability, indeed, may be subject to attacks based on object-level and/or meta-level background knowledge. This behavior makes such a property desirable in CQE, and is the one we guarantee with the notions of censors studied in this thesis.

Our investigation, however, differs from previous ones in various aspects. Compared to the initial work on CQE, such as [108, 31, 32, 34], conducted in the context of databases, our research addresses this problem in the ambit of Description Logic ontologies and Semantic Web applications, similarly as [112, 37, 57, 58]. CQE in this scenario is particularly challenging, considered that the usage of ontologies enables the deduction of implicit information from explicit data, which further escalates the risk of information leakage.

In this respect, the latter references are closer to our work. In more detail, reference [112] defines CQE for ontologies specified in Boolean $\mathcal{ALC}$ and queries that are $\mathcal{ALC}$ formulas. The paper identifies useful and reasonable properties that a censor should have for protecting confidentiality. In [57] the authors study CQE for ontologies in OWL 2 RL, one of the tractable profiles of OWL 2 [97], and a policy expressed by a set of ground atoms. One of the main outcomes of this paper is the identification of a subset of OWL 2 RL for which the computation of a censor is tractable. It has to be noted that the policy language considered in the two works above is not able to consider protection rules involving CQs, as we do. The complexity of censor computation is also studied in [58], where the same authors extend the framework of [57] to consider ontologies specified either in Datalog or in any of the OWL 2 profiles, with the policy expressed as a single CQ.

It is worthwhile remarking that all papers we mentioned so far focus on the problem of verifying the existence of a (optimal) censor ensuring a secure view of the data and on how to construct it. This approach is similar in spirit to what we discuss in Section 3.2, i.e. the strategy of materializing one GA censor among several possible optimal ones and then using exclusively such a censor for query answering. Whereas on the one hand picking up an optimal censor guarantees confidentiality protection, on the other hand the selection is typically arbitrary, if there is no additional metadata supporting such a choice. To overcome this problem, in the present thesis we also consider a different strategy that involves reasoning skeptically with respect to all optimal censors (Definition 7). This alternative approach was already briefly discussed in [57] and then elaborated and deeply investigated in [83]. In more detail, the latter paper generalizes the notion of censor given in [57, 58], by introducing the so-called censor language as a parameter. This idea is recalled here in Definition 4. Then, [83] studies entailment of CQs under GA censors and censors in **CQ** (cf. Definition 7) from the computational viewpoint, for ontologies in $\mathcal{EL}_\bot$ and DL-Lite$_\mathcal{R}$, the logics at the basis of the OWL 2 profiles OWL 2 EL and OWL 2 QL, respectively, and for policies specified in terms of CQs, analogously to the present thesis. Indistinguishability is however not considered (this aspect is also overlooked in [57, 58]). The notion of indistinguishability has also been proposed as built-in feature of censors, as in [52]. In this work, the authors study the problem of skeptical reasoning under such *indistinguishability-based (IB) censors* and a sound approximation thereof, called *quasi-optimal IB censor*. It can be shown that such an approximation corresponds to the notion of IGA censor, which however enjoys indistinguishability as an incidental property rather than by explicit definition.

As discussed in the previous sections, besides enjoying indistinguishability (Proposition 6), IGA censors have a desirable computational behaviour, allowing for conjunctive query answering in AC^0 in data complexity (Corollary 1). On the other hand, an IGA censor in general conceals more data than other forms of censors (e.g., GA censors) to preserve confidentiality. Equipping ontologies with preferences between predicates expressing priorities on the way in which data should be censored may help to mitigate this last issue, as we discussed in Chapter 4.

To the best of our knowledge, our research represents the first investigation into CQE within prioritized ontologies. The priority-based semantics that we propose are adapted from the literature on CQA (as said in the introduction, an in-depth investigation of the connection between the CQA and CQE frameworks can be

found in [83]). In the context of CQA, the use of preferences originates in [111] to select a set of preferred repairs, thus reducing the level of non-determinism in reasoning. The setting of preferred repairs has then been investigated in several other papers, in the context of both databases (see, e.g., [67, 74]) and ontologies (see, e.g., [26, 25, 39, 90]).

Among the mentioned papers, the work [25] is certainly the closest to our research. Indeed, our DD censor has a correspondence with the grounded extension introduced in [25] through a transformation of the CQA problem into an argumentation framework. Besides the differences between the two considered settings, we remark that priorities considered in [25] are specified between ABox facts, whereas we here assume priorities between ontology predicates, maintaining this aspect at the intensional level, and thus making priorities easier to manage from the modeling viewpoint. Moreover, our treatment is tailored to CQE, and does not require transformation into a different problem, thus streamlining the technical aspects of the approach.

It is also worthwhile to mention reference [22], which studies data privacy in the context of databases. In that work, the policy is expressed by means of secrecy views, i.e. views on the database specifying what to hide, and it is enforced by means of virtual updates of the database where attribute values are minimally changed with SQL nulls.

We now briefly discuss the literature related to the experimental part. Despite being suited for data sharing and interoperability, existing OBDA solutions do not provide any explicit support for the protection of confidential data, and the research has so far produced only initial theoretical contributions toward managing privacy issues within the OBDA paradigm. In [19, 18], the authors study the problem of determining whether information that is declared confidential at the sources through a protection policy, as in CQE, can be inferred by a user only on the basis of the data she can access through the OBDA system, assuming that she is knowledgeable about the OBDA specification. The paper studies the computational complexity of the above problem, under various assumptions on the forms of the mapping (GAV or LAV) and on the complexity of conjunctive query entailment for the given ontology language. Both the above paper and our work focus on the role of an OBDA mapping in filtering data coming from the sources with respect to some data protection requirements specified in a declarative way. However, we consider policies over the TBox of the OBDA specification and look at the mapping as a means to protect data, whereas in [19] policies are specified at the source level and the mapping is seen as a potential cause for revealing secrets. Data-independent compliance based on source indistinguishability (defined as the problem of deciding whether a policy p admits a disclosure) has been then recently studied in [16] in the presence of various forms of TGDs specified over the sources. This paper however does not specifically consider OBDA, but carries out its investigation within a virtual data integration framework (similar to OBDA, but with the ontology substituted by a relational schema, called *global schema*), and considers simple forms of mappings and a global schema without constraints. Potential disclosure of information declared confidential at the level of the source schema has been also studied in [99, 96, 49], in the context of data integration and exchange, also in these cases when the global schema is a flat relational schema, and not a logical theory expressed through ontological

axioms as in OBDA.

10.2 Consistent query answering

CQA was originally proposed for relational databases by Arenas *et al.* in [7], which introduced the notions of repairs and consistent answers, and repairs were intended as the ones minimizing the symmetric difference with the original database. Ever since, many works considered various kinds of integrity constraints and chose different repairing approaches. Other types of repairs include the maximal consistent subsets of the original database [51], as well as the minimal consistent supersets [114]. The notion of maximality is also sometimes based on the cardinality [87]. Here we define a repair as the maximal consistent superset of the original database, i.e. we adopt the so-called tuple-deletion semantics.

For what concerns the problem of repair checking, the most relevant papers that are strictly related to our investigation are [51, 2, 110, 72, 114], which deeply explored the problem for many classes of dependencies.

Also, the problem of finding consistent answers for a given query under tuple-deletion semantics has been extensively studied by [51] and [114], with a particular focus on (some forms of) tuple-generating dependencies. As shown in the conclusions, some of our results extend the ones provided in these two papers. Special attention should also be paid to [70] and [78], which presented FO-rewritable techniques for solving the CQA problem, though limiting the set of integrity constraints to (primary) key dependencies and making some assumptions on the user query.

In the mentioned works, all of which best reviewed in [50, 21, 119], the most common entailment semantics adopted corresponds to the one that here we call AR-entailment. The IAR semantics, instead, was previously studied in the context of ontologies by [80], and further investigated for multiple DL languages in [81, 105, 82]. In particular, the latter proved IAR-entailment of CQs to be FO rewritable for the language DL-Lite$_{\mathcal{R},den}$. Afterward, reference [24] proposed a new semantics named ICR ("intersection of closed repairs"), which IAR is a sound approximation of, showing its FO expressibility for simple ontologies.

Both the entailment problems that we consider have also been studied in the field of existential rules [92, 93, 91, 89], and systematically revised in [88]. However, all such results are not straightforwardly transposable under CWA, which is our case study. Indeed, in this thesis we aim to provide a complementary analysis to the "open" case, and we also give some first contributions on how to mix the two scenarios together.

Outside the context of inconsistency-tolerant query answering, the integration between OWA and CWA has already been proposed in various forms, especially for reasoning with Description Logics KBs. The most common approach,—that here we follow—consists in distinguishing between open and closed predicates. In the literature of DLs, this idea was encoded in the notion of DBox [106, 69], i.e. a set of assertions involving closed predicates that, when coupled with a DL TBox, allows to perform such kind of reasoning. More recently, it has also been considered for ontology-mediated query answering, both when the intensional knowledge is expressed in some DL [94, 95, 5, 48] or in some class of existential rules [28]. In [15,

17], closed and open predicates take the form of *visible* and *invisible* relations of arbitrary arity, respectively, and the end user can pose BUCQs (or their negation) over the visible part of the datasource. In [6], the authors proposed to treat part of the variables occurring in rules as "closed", i.e. assignable only by choosing among a finite set of "known" values.

Still other well-known approaches consist in using epistemic operators [73, 44], or considering hybrid knowledge bases [103, 104, 65, 98, 76, 13]. The latter are formalisms for integrating a classic Description Logics KB with a set of Datalog rules (or some extension thereof). More recently, in [71] the authors studied how to reason with existential rules under stable model semantics, thus addressing the problem of mixing the classical (open) reasoning paradigm with the non-monotonic (closed) one.

To the best of our knowledge, the sole research resembling CQA with open and closed predicates is the one conducted in [3] and [4], where the authors examined the problem of explaining SHACL non-validation of RDF graphs. SHACL [117] is a recent language recommended by W3C used for expressed constraints on *shapes* which an RDF graph must conform to. The problems considered in that paper, however, are technically not comparable with the ones studied in the present thesis, because the language used for expressing constraints is very different. Moreover, the *explanation* (which could be brought back to the notion of repair in CQA) may also contain nodes (i.e. facts) not belonging to the original graph (i.e. the database), and they do not consider the problem of query answering over repairs, except for a case similar to the problem of instance checking examined in our research.

Chapter 11

Conclusions

In the first part of this doctoral thesis, we investigated controlled query evaluation in Description Logics. In particular, we studied different types of censors and identified the ones satisfying the desirable property of indistinguishability. We also introduced an intersection-based semantics for BUCQs entailment that soundly approximates skeptical reasoning while enjoying first-order rewritability in the case of DL-Lite$_\mathcal{R}$ ontologies. We then enriched the framework with the possibility of specifying a priority relation over the predicate signature, and presented a well-founded entailment semantics that improves the throughput of query answers while still preserving confidentiality. At the same time, this novel approach retains the indistinguishability property and maintains the favorable computational complexity of the intersection-based scenario.

A possible improvement for our prioritized framework may consist in enriching the preference mechanism, e.g. adapting to the CQE setting one of the recent approaches to preferred repairs in CQA [26, 39, 90]. Also, we are currently studying the problem of intensionally (i.e. independently of the ABox) deciding whether, for a given CQE specification, there exists an integer k such that the k-DD censor converges on the DD censor. Interestingly, for such specifications, entailment of BUCQs under the DD censor is first-order rewritable.

Our experiments on prioritized CQE, conducted in the OBDA setting, show applicability in the practice of our technique, and how priorities, besides being an important modeling feature for the designer, play an important role in increasing the amount of (policy-safe) answers disclosed to the user.

We remark that, throughout this thesis, we employed the unique name assumption (UNA). It is important to note, however, that such an assumption can be dropped for all the results concerning DL-Lite$_\mathcal{R}$, which is well-known to be insensitive to its adoption for CQ answering [9] (i.e., given a DL-Lite$_\mathcal{R}$ ontology $\mathcal{O}$ and a BUCQ q, we have that q evaluates to true in $\mathcal{O}$ under UNA if and only if it does without UNA), and the same applies to the entailment problems that we examined. On the other hand, when TBoxes are expressed in DL-Lite$_\mathcal{A}$ such an assumption becomes necessary in order not to loose its nice computational properties [42]. Furthermore, the UNA must be adopted whenever dealing with policies or dependencies involving inequalities.

We furthered our exploration on CQE by presenting a maximally cooperative

approach that, for BUCQs, is computationally not harder than the previous static and less cooperative approaches to CQE. Moreover, we have defined a new query rewriting algorithm to solve the query entailment problem in this framework. However, more work is still needed to define a practical query answering technique and to extend it to non-Boolean UCQs. Indeed, unlike for the foundational and prioritized CQE, the methodology tailored for the dynamic scenario can not be readily transposed to the case of non-Boolean queries. We can show an intuition of this issue by means of the following example.

Example 46. Consider the CQE specification $\langle \mathcal{T}, \mathcal{P} \rangle$ of Example 34, and suppose the first query posed to the system is $q_1(x, y) = \mathsf{buy}(x, y)$. Since no other query precedes q_1, we can safely reveal all the ground substitutions σ of its free variables such that $\sigma(q_1)$ is entailed by $\mathcal{T} \cup \mathcal{A}$, as $\sigma(q_1)$ *per se* can not be used for inferring any secret. Let us now take the query $q_2(x) = \mathsf{Antiseizure}(x)$. The answers that can be returned for this second query now strictly depend on the first answer set. For instance, if the underlying ABox was as in Example 34, then the user would know that $\mathsf{buy}(\mathsf{john}, \mathsf{m}_a)$ is true, so we should not reveal also $\mathsf{Antiseizure}(\mathsf{m}_a)$. The opposite could occur with an ABox not containing the fact $\mathsf{buy}(\mathsf{john}, \mathsf{m}_a)$.

Intuitively, it might be the case that it is impossible to rewrite q_2 in order to maximize the answer set and at the same time preserving confidentiality without knowing which answers have been returned for the first query. In other words, it might not be possible to build an intensional FO rewriting of q_2. □

Our current research is also focused on this latest aspect, as we believe it is very interesting research direction that merits further investigation.

Overall, as for future work, it would be highly intriguing to conduct a computational analysis involving ontologies specified in alternative Description Logic fragments.

The second part of the thesis falls within the scope of research on consistent query answering. We employ a notion of repair based on tuple-deletion, and consistency of databases is checked with respect to disjunctive embedded dependencies with inequalities, a very expressive language for schema constraints. We studied different decision problems related to the notion of database repair under tuple-deletion semantics for the whole class of DED$^{\neq}$s, for the linear, acyclic, full, sticky and guarded subclasses, and for all the possible combinations thereof. We have shown tractability with respect to data complexity of the examined decision problems for several of such classes of dependencies. We then relaxed the closed-world assumption for part of the predicate signature, which instead is considered "open". This kind of combination of OWA and CWA is usually very challenging from the computational viewpoint. Nonetheless, we identified a practical condition which, by limiting the interaction between open and closed predicates, allows to mitigate the computational complexity in this scenario, while preserving the possibility of having (closed-head) dependencies with disjunctive-heads and inequalities.

In our opinion, the most important part of our findings is the identification of many classes of existential rules in which the problems studied are in AC^0 in data complexity. On the one hand, this is a bit surprising, since consistent query answering is generally known to be a computationally hard task. On the other hand,

for the problems in AC^0 we have actually shown that such problems can be solved by the evaluation of an FO sentence over the database: therefore, these results (and in particular Definition 37, Definition 38, Definition 39 and Definition 40) can be a starting point towards the development of practical algorithms for consistent query answering based on FO rewriting techniques.

Looking forward, we are eager to delve deeper into our research on the partially-complete semantics, especially with regard to dependencies going beyond the OCS case. Finally, due to the close relationship of these problems with CQE, we are very interested in investigating the consequences of our results in the context of data protection.

Bibliography

[1] Abiteboul, S., Hull, R., and Vianu, V. *Foundations of Databases*. Addison Wesley Publ. Co. (1995).

[2] Afrati, F. N. and Kolaitis, P. G. Repair checking in inconsistent databases: Algorithms and complexity. In *Proc. of the 12th Int. Conf. on Database Theory (ICDT)*, vol. 361 of *ACM Int. Conf. Proc. Series*, pp. 31–41. ACM Press (2009).

[3] Ahmetaj, S., David, R., Ortiz, M., Polleres, A., Shehu, B., and Šimkus, M. Reasoning about explanations for non-validation in SHACL. In *Proc. of the 18th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pp. 12–21 (2021).

[4] Ahmetaj, S., David, R., Polleres, A., and Šimkus, M. Repairing SHACL constraint violations using answer set programming. In *Proc. of the 21st Int. Semantic Web Conf. (ISWC)*, pp. 375–391. Springer, Cham (2022).

[5] Ahmetaj, S., Ortiz, M., and Simkus, M. Polynomial rewritings from expressive description logics with closed predicates to variants of datalog. *Artificial Intelligence*, **280** (2020), 103220.

[6] Amendola, G., Leone, N., Manna, M., and Veltri, P. Enhancing existential rules by closed-world variables. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, p. 1676–1682. AAAI Press (2018).

[7] Arenas, M., Bertossi, L., and Chomicki, J. Consistent query answers in inconsistent databases. In *Proc. of the 18th ACM SIGMOD SIGACT SIGART Symp. on Principles of Database Systems (PODS)*, p. 68–79. Association for Computing Machinery (1999).

[8] Arming, S., Pichler, R., and Sallinger, E. Complexity of Repair Checking and Consistent Query Answering. In *Proc. of the 19th Int. Conf. on Database Theory (ICDT)*, vol. 48 of *Leibniz Int. Proc. in Informatics (LIPIcs)*, pp. 21:1–21:18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2016).

[9] Artale, A., Calvanese, D., Kontchakov, R., and Zakharyaschev, M. The DL-Lite family and relations. *J. of Artificial Intelligence Research*, **36** (2009), 1.

[10] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. F. (eds.). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edn. (2007).

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

[12] Baget, J., Leclère, M., Mugnier, M., and Salvat, E. On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, **175** (2011), 1620.

[13] Bajraktari, L., Ortiz, M., and Šimkus, M. Combining rules and ontologies into clopen knowledge bases. *Proc. of the 32nd AAAI Conf. on Artificial Intelligence (AAAI)*, **32** (2018).

[14] Bayardo Jr., R. J. and Agrawal, R. Data privacy through optimal k-anonymization. In *Proc. of the 31st IEEE Int. Conf. on Data Engineering (ICDE)*, pp. 217–228. IEEE Computer Society Press (2005).

[15] Benedikt, M., Bourhis, P., Cate, B. t., and Puppis, G. Querying visible and invisible information. In *Proc. of the 31st IEEE Symp. on Logic in Computer Science (LICS)*, pp. 1–10 (2016).

[16] Benedikt, M., Bourhis, P., Jachiet, L., and Thomazo, M. Reasoning about disclosure in data integration in the presence of source constraints. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1551–1557 (2019).

[17] Benedikt, M., Bourhis, P., ten Cate, B., Puppis, G., and Vanden Boom, M. Inference from visible information and background knowledge. *ACM Trans. on Computational Logic*, **22** (2021), 13:1.

[18] Benedikt, M., Cuenca Grau, B., and Kostylev, E. Source information disclosure in ontology-based data integration. *Proc. of the 31st AAAI Conf. on Artificial Intelligence (AAAI)*, **31** (2017).

[19] Benedikt, M., Cuenca Grau, B., and Kostylev, E. V. Logical foundations of information disclosure in ontology-based data integration. *Artificial Intelligence*, **262** (2018), 52.

[20] Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web. *Scientific American*, **284** (2001), 34.

[21] Bertossi, L. E. Database repairs and consistent query answering: Origins and further developments. In *Proc. of the 38rd ACM SIGMOD SIGACT SIGAI Symp. on Principles of Database Systems (PODS)*, pp. 48–58. ACM Press (2019).

[22] Bertossi, L. E. and Li, L. Achieving data privacy through secrecy views and null-based virtual updates. *IEEE Trans. on Knowledge and Data Engineering*, **25** (2013), 987.

[23] Bhanot, R. and Hans, R. A review and comparative analysis of various encryption algorithms. *Int. J. of Security and Its Applications*, **9** (2015), 289.

[24] Bienvenu, M. On the complexity of consistent query answering in the presence of simple ontologies. In *Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI)*. AAAI Press (2012).

[25] Bienvenu, M. and Bourgaux, C. Querying and repairing inconsistent prioritized knowledge bases: Complexity analysis and links with abstract argumentation. In *Proc. of the 17th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pp. 141–151 (2020).

[26] Bienvenu, M., Bourgaux, C., and Goasdoué, F. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *Proc. of the 28th AAAI Conf. on Artificial Intelligence (AAAI)* (edited by C. E. Brodley and P. Stone), pp. 996–1002. AAAI Press (2014).

[27] Bienvenu, M., Bourgaux, C., and Goasdoué, F. Computing and explaining query answers over inconsistent DL-Lite knowledge bases. *J. of Artificial Intelligence Research*, **64** (2019), 563.

[28] Bienvenu, M. and Bourhis, P. Mixed-world reasoning with existential rules under active-domain semantics. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, p. 1558–1565. AAAI Press (2019).

[29] Biskup, J. For unknown secrecies refusal is better than lying. *Data and Knowledge Engineering*, **33** (2000), 1.

[30] Biskup, J. and Bonatti, P. A. Lying versus refusal for known potential secrets. *Data and Knowledge Engineering*, **38** (2001), 199.

[31] Biskup, J. and Bonatti, P. A. Controlled query evaluation for enforcing confidentiality in complete information systems. *Int. J. of Information Security*, **3** (2004), 14.

[32] Biskup, J. and Bonatti, P. A. Controlled query evaluation for known policies by combining lying and refusal. *Ann. of Mathematics and Artificial Intelligence*, **40** (2004), 37.

[33] Biskup, J. and Bonatti, P. A. Controlled query evaluation with open queries for a decidable relational submodel. *Ann. of Mathematics and Artificial Intelligence*, **50** (2007), 39.

[34] Biskup, J. and Weibert, T. Keeping secrets in incomplete databases. *Int. J. of Information Security*, **7** (2008), 199.

[35] Bonatti, P. A. A false sense of security. *Artificial Intelligence*, **310** (2022).

[36] Bonatti, P. A., De Capitani di Vimercati, S., and Samarati, P. An algebra for composing access control policies. *Int. J. of Information Security*, **5** (2002), 1.

[37] Bonatti, P. A. and Sauro, L. A confidentiality model for ontologies. In *Proc. of the 12th Int. Semantic Web Conf. (ISWC)*, vol. 8218 of *Lecture Notes in Computer Science*, pp. 17–32. Springer (2013).

[38] Buss, S. R. Chapter II - First-order proof theory of arithmetic. In *Handbook of Proof Theory*, vol. 137 of *Studies in Logic and the Foundations of Mathematics*, pp. 79–147. Elsevier (1998).

[39] Calautti, M., Greco, S., Molinaro, C., and Trubitsyna, I. Preference-based inconsistency-tolerant query answering under existential rules. *Artificial Intelligence*, **312** (2022), 103772.

[40] Calì, A., Gottlob, G., and Lukasiewicz, T. A general Datalog-based framework for tractable query answering over ontologies. *J. of Web Semantics*, **14** (2012), 57.

[41] Calì, A., Gottlob, G., and Pieris, A. Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence*, **193** (2012), 87.

[42] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., and Rosati, R. Ontologies and databases: The DL-Lite approach. In *Reasoning Web. Semantic Technologies for Informations Systems – 5th Int. Summer School Tutorial Lectures (RW)* (edited by S. Tessaris and E. Franconi), vol. 5689 of *Lecture Notes in Computer Science*, pp. 255–356. Springer (2009).

[43] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI)*, pp. 602–607 (2005).

[44] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. EQL-Lite: Effective first-order query processing in description logics. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 274–279. Morgan Kaufmann Publishers Inc. (2007).

[45] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning*, **39** (2007), 385.

[46] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. Data complexity of query answering in description logics. *Artificial Intelligence*, **195** (2013), 335.

[47] Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R., and Vetere, G. DL-Lite: Practical reasoning for rich DLs. In *Proc. of the 17th Int. Workshop on Description Logic (DL)*, vol. 104 of *CEUR Electronic Workshop Proc.,* `http://ceur-ws.org/` (2004).

[48] Cauli, C., Ortiz, M., and Piterman, N. Closed- and open-world reasoning in DL-Lite for cloud infrastructure security. In *Proc. of the 18th Int.*

Conf. on the Principles of Knowledge Representation and Reasoning (KR), pp. 174–183 (2021).

[49] Chirkova, R. and Yu, T. Exact detection of information leakage: Decidability and complexity. *Trans. on Large-Scale Data- and Knowledge-Centered Systems*, **32** (2017), 1.

[50] Chomicki, J. Consistent query answering: Five easy pieces. In *Proc. of the 11th Int. Conf. on Database Theory (ICDT)*, vol. 4353 of *Lecture Notes in Computer Science*, pp. 1–17. Springer (2007).

[51] Chomicki, J. and Marcinkowski, J. Minimal-change integrity maintenance using tuple deletions. *Information and Computation*, **197** (2005), 90.

[52] Cima, G., Lembo, D., Rosati, R., and Savo, D. F. Controlled query evaluation in description logics through instance indistinguishability. In *Proc. of the 29th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1791–1797 (2020).

[53] Console, M., Mora, J., Rosati, R., Santarelli, V., and Savo, D. F. Effective computation of maximal sound approximations of description logic ontologies. In *Proc. of the 13th Int. Semantic Web Conf. (ISWC)*, vol. 8797 of *Lecture Notes in Computer Science*, pp. 164–179. Springer (2014).

[54] Cuenca Grau, B. Privacy in ontology-based information systems: A pending matter. *Semantic Web J.*, **1** (2010), 137.

[55] Cuenca Grau, B. and Horrocks, I. Privacy-preserving query answering in logic-based information systems. In *Proc. of the 18th Eur. Conf. on Artificial Intelligence (ECAI)*, pp. 40–44 (2008).

[56] Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., and Sattler, U. OWL 2: The next step for OWL. *J. of Web Semantics*, **6** (2008), 309.

[57] Cuenca Grau, B., Kharlamov, E., Kostylev, E. V., and Zheleznyakov, D. Controlled query evaluation over OWL 2 RL ontologies. In *Proc. of the 12th Int. Semantic Web Conf. (ISWC)*, vol. 8218 of *Lecture Notes in Computer Science*, pp. 49–65. Springer (2013).

[58] Cuenca Grau, B., Kharlamov, E., Kostylev, E. V., and Zheleznyakov, D. Controlled query evaluation for Datalog and OWL 2 profile ontologies. In *Proc. of the 24th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 2883–2889 (2015).

[59] Cuenca Grau, B. and Kostylev, E. V. Logical foundations of linked data anonymisation. *J. of Artificial Intelligence Research*, **64** (2019), 253.

[60] Das, S., Sundara, S., and Cyganiak, R. R2RML: RDB to RDF mapping language. W3C Recommendation, World Wide Web Consortium (2012). Available at `http://www.w3.org/TR/r2rml/`.

[61] De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R., Ruzzi, M., and Savo, D. F. MASTRO: A reasoner for effective Ontology-Based Data Access. In *Proc. of the 1st Int. Workshop on OWL Reasoner Evaluation (ORE)* (2012).

[62] Deutsch, A. FOL modeling of integrity constraints (dependencies). In *Encyclopedia of Database Systems, Second Edition*. Springer (2018).

[63] Deutsch, A. and Tannen, V. XML queries and constraints, containment and reformulation. *Theoretical Computer Science*, **336** (2005), 57.

[64] Doan, A., Halevy, A. Y., and Ives, Z. G. *Principles of Data Integration*. Morgan Kaufmann (2012). ISBN 978-0-12-416044-6.

[65] Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., and Tompits, H. Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence*, **172** (2008), 1495.

[66] European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016. *Official J. of the European Union*, **L 119** (2016), 55.

[67] Fagin, R., Kimelfeld, B., and Kolaitis, P. G. Dichotomies in the complexity of preferred repairs. In *Proc. of the 34rd ACM SIGMOD SIGACT SIGAI Symp. on Principles of Database Systems (PODS)*, pp. 3–15 (2015).

[68] Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. Data exchange: Semantics and query answering. *Theoretical Computer Science*, **336** (2005), 89.

[69] Franconi, E., Ibáñez-García, Y. A., and Seylan, I. Query answering with DBoxes is hard. In *Proc. of the 7th Workshop on Methods for Modalities, M4M 2011, and the 4th Workshop on Logical Aspects of Multi-Agent Systems, LAMAS 2011*, vol. 278 of *Electronic Notes in Theoretical Computer Science*, pp. 71–84. Elsevier (2011).

[70] Fuxman, A. and Miller, R. J. First-order query rewriting for inconsistent databases. In *Proc. of the 10th Int. Conf. on Database Theory (ICDT)*, vol. 3363 of *Lecture Notes in Computer Science*, pp. 337–351. Springer (2005).

[71] Gottlob, G., Hernich, A., Kupke, C., and Lukasiewicz, T. Stable model semantics for guarded existential rules and description logics: Decidability and complexity. *J. of the ACM*, **68** (2021), 35:1.

[72] Grahne, G. and Onet, A. Data correspondence, exchange and repair. In *Proc. of the 13th Int. Conf. on Database Theory (ICDT)*, ACM Int. Conf. Proc. Series, pp. 219–230. ACM Press (2010).

[73] Grimm, S. and Motik, B. Closed world reasoning in the Semantic Web through epistemic operators. In *Proc. of the 1st Int. Workshop on OWL: Experiences and Directions (OWLED)*, vol. 188 of *CEUR Electronic Workshop Proc.,* `http://ceur-ws.org/` (2005).

[74] Kimelfeld, B., Livshits, E., and Peterfreund, L. Counting and enumerating preferred database repairs. *Theoretical Computer Science*, **837** (2020), 115.

[75] Klyne, G. and Carroll, J. J. Resource Description Framework (RDF): Concepts and abstract syntax. W3C Recommendation, World Wide Web Consortium (2004). Available at `http://www.w3.org/TR/rdf-concepts/`.

[76] Knorr, M., Alferes, J. J., and Hitzler, P. Local closed world reasoning with description logics under the well-founded semantics. *Artificial Intelligence*, **175** (2011), 1528.

[77] König, M., Leclère, M., Mugnier, M., and Thomazo, M. Sound, complete and minimal UCQ-rewriting for existential rules. *Semantic Web J.*, **6** (2015), 451.

[78] Koutris, P. and Wijsen, J. Consistent query answering for self-join-free conjunctive queries under primary key constraints. *ACM Trans. on Database Systems*, **42** (2017), 9:1.

[79] Lanti, D., Rezk, M., Xiao, G., and Calvanese, D. The NPD benchmark: Reality check for OBDA systems. In *Proc. of the 18th Int. Conf. on Extending Database Technology (EDBT)*, pp. 617–628 (2015).

[80] Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., and Savo, D. F. Inconsistency-tolerant semantics for description logics. In *Proc. of the 4th Int. Conf. on Web Reasoning and Rule Systems (RR)*, vol. 6333 of *Lecture Notes in Computer Science*, pp. 103–117. Springer (2010).

[81] Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., and Savo, D. F. Query rewriting for inconsistent DL-Lite ontologies. In *Proc. of the 5th Int. Conf. on Web Reasoning and Rule Systems (RR)*, vol. 6902 of *Lecture Notes in Computer Science*, pp. 155–169. Springer (2011).

[82] Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., and Savo, D. F. Inconsistency-tolerant query answering in ontology-based data access. *J. of Web Semantics*, **33** (2015), 3.

[83] Lembo, D., Rosati, R., and Savo, D. F. Revisiting controlled query evaluation in description logics. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 1786–1792 (2019).

[84] Lenzerini, M. Data integration: A theoretical perspective. In *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, p. 233–246. Association for Computing Machinery, New York, NY, USA (2002). ISBN 1581135076.

[85] Lenzerini, M. Managing data through the lens of an ontology. *AI Magazine*, **39** (2018), 65.

[86] Lloyd, J. W. *Foundations of Logic Programming (Second, Extended Edition)*. Springer, Berlin, Heidelberg (1987).

[87] Lopatenko, A. and Bertossi, L. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *Proc. of the 11th Int. Conf. on Database Theory (ICDT)*, pp. 179–193. Springer (2006).

[88] Lukasiewicz, T., Malizia, E., Martinez, M. V., Molinaro, C., Pieris, A., and Simari, G. I. Inconsistency-tolerant query answering for existential rules. *Artificial Intelligence*, **307** (2022).

[89] Lukasiewicz, T., Malizia, E., and Molinaro, C. Complexity of approximate query answering under inconsistency in Datalog±. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, p. 1921–1927. AAAI Press (2018).

[90] Lukasiewicz, T., Malizia, E., and Molinaro, C. Complexity of inconsistency-tolerant query answering in datalog+/- under preferred repairs. In *Proc. of the 20th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pp. 472–481 (2023).

[91] Lukasiewicz, T., Martinez, M. V., Pieris, A., and Simari, G. I. From classical to consistent query answering under existential rules. *Proc. of the 29th AAAI Conf. on Artificial Intelligence (AAAI)*, **29** (2015), 1546.

[92] Lukasiewicz, T., Martinez, M. V., and Simari, G. I. Inconsistency handling in Datalog+/- ontologies. In *Proc. of the 20th Eur. Conf. on Artificial Intelligence (ECAI)*, vol. 242 of *Frontiers in Artificial Intelligence and Applications*, pp. 558–563. IOS Press (2012).

[93] Lukasiewicz, T., Martinez, M. V., and Simari, G. I. Inconsistency-tolerant query rewriting for linear Datalog+/-. In *Proc. of the 2nd Int. Conf. on Datalog in Academia and Industry*, p. 123–134. Springer (2012).

[94] Lutz, C., Seylan, I., and Wolter, F. Ontology-based data access with closed predicates is inherently intractable (sometimes). In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, p. 1024–1030. AAAI Press (2013).

[95] Lutz, C., Seylan, I., and Wolter, F. The data complexity of ontology-mediated queries with closed predicates. *Logical Methods in Computer Science*, **15** (2019).

[96] Miklau, G. and Suciu, D. A formal analysis of information disclosure in data exchange. *J. of Computer and System Sciences*, **73** (2007), 507.

[97] Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., and Lutz, C. OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, World Wide Web Consortium (2012). Available at `http://www.w3.org/TR/owl2-profiles/`.

[98] Motik, B. and Rosati, R. Reconciling description logics and rules. *J. of the ACM*, **57** (2010).

[99] Nash, A. and Deutsch, A. Privacy in GLAV information integration. In *Proc. of the 11th Int. Conf. on Database Theory (ICDT)*, pp. 89–103 (2007).

[100] Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., and Rosati, R. Linking data to ontologies. *J. on Data Semantics*, **X** (2008), 133.

[101] Prud'hommeaux, E. and Seaborne, A. SPARQL query language for RDF. W3C Recommendation, World Wide Web Consortium (2008). Available at `http://www.w3.org/TR/rdf-sparql-query`.

[102] Reiter, R. Towards a logical reconstruction of relational database theory. In *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*. Springer (1984).

[103] Rosati, R. On the decidability and complexity of integrating ontologies and rules. *J. of Web Semantics*, **3** (2005), 61.

[104] Rosati, R. DL+log: Tight integration of description logics and disjunctive Datalog. In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*, pp. 68–78 (2006).

[105] Rosati, R. On the complexity of dealing with inconsistency in description logic ontologies. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, vol. 2, p. 1057–1062. AAAI Press (2011).

[106] Seylan, I., Franconi, E., and de Bruijn, J. Effective query rewriting with ontologies over DBoxes (extended abstract). In *Proc. of the 22nd Int. Workshop on Description Logic (DL)*, vol. 477 of *CEUR Electronic Workshop Proc.,* `http://ceur-ws.org/` (2009).

[107] Shostack, A. *Threat Modeling: Designing for Security*. Wiley (2014).

[108] Sicherman, G. L., de Jonge, W., and van de Riet, R. P. Answering queries without revealing secrets. *ACM Trans. on Database Systems*, **8** (1983), 41.

[109] Skjæveland, M. G., Lian, E. H., and Horrocks, I. Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web data. In *Proc. of the 12th Int. Semantic Web Conf. (ISWC)*, vol. 8219 of *Lecture Notes in Computer Science*, p. 162–177. Springer (2013).

[110] Staworko, S. and Chomicki, J. Consistent query answers in the presence of universal constraints. *Information Systems*, **35** (2010), 1.

[111] Staworko, S., Chomicki, J., and Marcinkowski, J. Prioritized repairing and consistent query answering in relational databases. *Ann. of Mathematics and Artificial Intelligence*, **64** (2012), 209.

[112] Studer, T. and Werner, J. Censors for Boolean description logic. *Trans. on Data Privacy*, **7** (2014), 223.

[113] Sweeney, L. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. on Uncertainty, Fuzziness and Knowledge-Based Systems*, **10** (2002), 571.

[114] ten Cate, B., Fontaine, G., and Kolaitis, P. G. On the data complexity of consistent query answering. In *Proc. of the 15th Int. Conf. on Database Theory (ICDT)*, pp. 22–33. ACM Press (2012).

[115] Topor, R. *Safety and Domain Independence*, pp. 3269–3273. Springer New York (2018).

[116] Vardi, M. Y. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC)*, pp. 137–146 (1982).

[117] W3C. *Shapes Constraint Language (SHACL)*. W3C Recommendation (20 July 2017). Available at `https://www.w3.org/TR/shacl/`.

[118] Wijsen, J. Database repairing using updates. *ACM Trans. on Database Systems*, **30** (2005), 722.

[119] Wijsen, J. Foundations of query answering on inconsistent databases. *SIGMOD Record*, **48** (2019), 6–16.

[120] Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., and Zakharyaschev, M. Ontology-based data access: A survey. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 5511–5519 (2018).

[121] Zhang, Z. and Mendelzon, A. Authorization views and conditional query containment. In *Proc. of the 10th Int. Conf. on Database Theory (ICDT)*, vol. 3363 of *Lecture Notes in Computer Science*, pp. 259–273. Springer (2005).

Appendix A

The *PerfectRef* algorithm

In this section we illustrate the algorithm *PerfectRef* (Algorithm 7), that is extensively used throughout the thesis. More precisely, we show a slightly different version from the original introduced in [45]. The procedure takes indeed as input a BUCQ instead of a CQ, and gives as output a BUCQ instead of a UCQ.

In the following, we call *shared* variable every variable occurring more than once in a BCQ. If a variable is non-shared, we call it *unbound*. Given a BCQ q, the function $\tau(q)$ returns a new BCQ where each occurrence of unbound variables in q is replaced with the special symbol _. Given a BCQ q and two atoms α (which occurs in q) and β, the function $replace(q, \alpha, \beta)$ returns the BCQ obtained by replacing α with β in q. Moreover, if α and β are unifiable and they both occur in q, the function $reduce(q, \alpha, \beta)$ returns the BCQ $\sigma(q)$, where σ is an MGU for α and β. Such an MGU σ considers each occurrence of the symbol _ as a distinct variable, and it never replaces a shared variable with such a symbol.

Given a DL-Lite$_\mathcal{R}$ TBox $\mathcal{T}$ and a predicate atom α, we say that a positive inclusion (PI) $I \in \mathcal{T}$ *is applicable to* α if one of the following conditions holds:

- $\alpha = C(x)$ for some $C \in \Gamma_C$, and C is in the right-hand side of I;
- $\alpha = P(x, _)$ for some $P \in \Gamma_R$, and $\exists P$ is in the right-hand side of I;
- $\alpha = P(_, y)$ for some $P \in \Gamma_R$, and $\exists P^-$ is in the right-hand side of I;
- $\alpha = P(x, y)$ for some $P \in \Gamma_R$, and P is in the right-hand side of I.

We then indicate with $gr(\alpha, I)$ the result of applying I to α. Formally:

- If $\alpha = A(x)$ and $I = A_1 \sqsubseteq A$, then $gr(\alpha, I) = A_1(x)$;
- If $\alpha = A(x)$ and $I = \exists P \sqsubseteq A$, then $gr(\alpha, I) = P(x, _)$;
- If $\alpha = A(x)$ and $I = \exists P^- \sqsubseteq A$, then $gr(\alpha, I) = P(_, x)$;
- If $\alpha = P(x, _)$ and $I = A \sqsubseteq \exists P$, then $gr(\alpha, I) = A(x)$;
- If $\alpha = P(x, _)$ and $I = \exists P_1 \sqsubseteq \exists P$, then $gr(\alpha, I) = P_1(x, _)$;
- If $\alpha = P(x, _)$ and $I = \exists P_1^- \sqsubseteq \exists P$, then $gr(\alpha, I) = P_1(_, x)$;

- If $\alpha = P(_, x)$ and $I = A \sqsubseteq \exists P^-$, then $gr(\alpha, I) = A(x)$;
- If $\alpha = P(_, x)$ and $I = \exists P_1 \sqsubseteq \exists P^-$, then $gr(\alpha, I) = P_1(x, _)$;
- If $\alpha = P(_, x)$ and $I = \exists P_1^- \sqsubseteq \exists P^-$, then $gr(\alpha, I) = P_1(_, x)$;
- If $\alpha = P(x, y)$ and $I \in \{P_1 \sqsubseteq P, P_1^- \sqsubseteq P^-\}$, then $gr(\alpha, I) = P_1(x, y)$;
- If $\alpha = P(x, y)$ and $I \in \{P_1^- \sqsubseteq P, P_1 \sqsubseteq P^-\}$, then $gr(\alpha, I) = P_1(y, x)$.

We can now describe Algorithm 7. As one can see, it considers each BCQ q occurring in the input BUCQ q' and it generates new BCQs by either (i) rewriting one atom at a time in each q (Steps 5-8) or (ii) unifying two atoms in q (Steps 9-11). Intuitively, point (i) is executing the so-called backward chaining (i.e. it applies positive inclusions from right to left for inferring new queries), whereas point (ii) can be necessary for "unlocking" new queries to be rewritten in the next iteration of the outermost loop.

Algorithm 7 *PerfectRef*

input: a BUCQ q', a TBox $\mathcal{T}$;
output: a BUCQ;
1: $\Phi \leftarrow \{q \mid q \in CQ(q')\}$
2: **repeat**
3: $\quad \Phi' \leftarrow \Phi$;
4: $\quad$ **for each** $q \in \Phi'$ **do**
5: $\quad\quad$ **for each** $\alpha \in predAt(q)$ **do**
6: $\quad\quad\quad$ **for each** PI $I \in \mathcal{T}$ **do**
7: $\quad\quad\quad\quad$ **if** I is applicable to α **then**
8: $\quad\quad\quad\quad\quad \Phi \leftarrow \Phi \cup \{replace(q, \alpha, gr(\alpha, I))\}$
9: $\quad\quad$ **for each** $\alpha, \beta \in predAt(q)$ **do**
10: $\quad\quad\quad$ **if** α and β are unifiable **then**
11: $\quad\quad\quad\quad \Phi \leftarrow \Phi \cup \{\tau(reduce(q, \alpha, \beta))\}$;
12: **until** $\Phi' = \Phi$;
13: **return** $\bigvee_{q \in \Phi} q$;

Appendix B

Complexity of decision problems in propositional logic

B.1 Introduction to propositional logic

Propositional logic is a branch of mathematical logic that deals with *propositional formulas* (or simply *propositions*), which are statements that are either true or false. The syntax of propositional logic involves the use of propositional variables, constants *true* and *false*, and logical connectives: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication).

Well-formed propositional formulas are defined as follows.

Definition 46. [Propositional formula] Let $\mathscr{X}$ be a set of propositional variables. Then:

- *true* and *false* are propositions;
- given a variable $x \in \mathscr{X}$, x is a proposition;
- given a proposition ϕ, $\phi \wedge \psi$, $\neg\phi$ is a proposition;
- given two propositions ϕ and ψ, $\phi \wedge \psi$, $\phi \vee \psi$ and $\phi \rightarrow \psi$ are propositions.

Every variable x and its negation $\neg x$ are also called *literals*. A disjunction of literals is called *clause*. If a propositional formula takes the form of a conjunction of clauses, we say that it is in *conjunctive normal form* (*CNF*). Given a formula ϕ, we write $PV(\phi)$ for denoting the set of variables occurring in ϕ. If ϕ is in CNF, we also use $clauses(\phi)$ for denoting the set of its clauses.

The semantics of propositional logic is given in terms of *interpretations* (or *valuations*) of propositional variables. A valuation of a set $\mathscr{X}$ of variables is a function assigning to each variable $x \in \mathscr{X}$ a *truth value* (either *true* or $false$). Given a proposition ϕ and a valuation ν for (a subset of) its variables, we write $\phi[\nu]$ for indicating the proposition obtained by replacing in ϕ each variable x in the domain of definition of ν with $\nu(x)$. For a proposition ϕ without variables, we recursively define its truth value as follows:

- if $\phi = true$, then ϕ is true;
- if $\phi = false$, then ϕ is false;

- if $\phi = \neg\psi$, then ϕ is true iff ψ is false, false otherwise;
- if $\phi = \psi_1 \wedge \psi_2$, then ϕ is true iff both ψ_1 and ψ_2 are true, false otherwise;
- if $\phi = \psi_1 \vee \psi_2$, then ϕ is true iff at least one between ψ_1 and ψ_2 is true, false otherwise;
- if $\phi = \psi_1 \rightarrow \psi_2$, then ϕ is true iff ψ_1 is false or ψ_2 is true, false otherwise;

If a valuation ν for the variables of ϕ is such that $\phi[\nu]$ is true, then we say that ν *satisfies* ϕ. When such a valuation exists, we say that ϕ is *satisfiable*.

A *quantified Boolean formula* (*QBF*) is a propositional formula every variable of which is quantified either with an existential ($\exists$) or a universal quantifier ($\forall$). We can extend the notion of truth value to QBFs as follows. A QBF ϕ is *valid* if and only if one of the following conditions holds:

- $\phi = \exists\mathbf{x}\,\psi(\mathbf{x})$[1] and, for some valuation ν of variables $\mathbf{x}$, $\psi[\nu]$ is valid;
- $\phi = \forall\mathbf{x}\,\psi(\mathbf{x})$ and, for every valuation ν of variables $\mathbf{x}$, $\psi[\nu]$ is valid;
- ϕ is a propositional formula and it is true.

We assume every QBF to be given in *prenex normal form*, i.e. the normal form where all the quantifiers are in the outermost scope of the formula.

B.2 Complexity of satisfiability and validity problems

B.2.1 CNF-SAT

The decision problem *CNF-SAT* involves determining whether a given propositional formula in CNF is satisfiable. The formal definition is as follows:

Definition 47. [CNF-SAT] Let ϕ be a formula in CNF. Does there exist a truth assignment that satisfies ϕ?

Proposition 37. *CNF-SAT is NP-complete.*

In the following, we illustrate some special cases which the above problem can specialize in, some of which have been proved to have a lower complexity.

B.2.2 3-SAT

A *3-CNF formula* is a formula in CNF where each clause contains at most three literals. The decision problem *3-SAT* involves determining whether a given propositional 3-CNF formula is satisfiable. The formal definition is as follows:

Definition 48. [3-SAT] Let ϕ be a 3-CNF formula ϕ. Does there exist a truth assignment that satisfies ϕ?

Proposition 38. *3-SAT is NP-complete.*

[1] Like for FO logic, we use the notation $\phi(\mathbf{x})$ for emphasizing all the variables $\mathbf{x}$ that occur not quantified in a subformula ϕ of a QBF.

B.2.3 Horn-CNF-SAT

A *Horn clause* is a disjunction of literals with at most one positive literal. It is also called *Horn rule* when it is given in the (equivalent) form $x \wedge \cdots \wedge y \rightarrow z$. The decision problem *HORN-CNF-SAT* involves determining whether a given CNF formula with Horn clauses is satisfiable. The formal definition is as follows:

Definition 49. [HORN-CNF-SAT] Let ϕ be a formula in CNF such that each $c \in \mathit{clauses}(\phi)$ is a Horn clause. Does there exist a truth assignment that satisfies ϕ?

Proposition 39. *HORN-CNF-SAT is PTIME-complete.*

B.2.4 Horn-3-SAT

A *Horn 3-CNF formula* is a 3-CNF formula consisting in a conjunction of Horn clauses. The decision problem *HORN-3-SAT* involves determining whether a given Horn 3-CNF formula is satisfiable. The formal definition is as follows:

Definition 50. [HORN-3-SAT] Let ϕ be a Horn 3-CNF formula. Does there exist a truth assignment that satisfies ϕ?

Proposition 40. *HORN-3-SAT is PTIME-complete.*

B.2.5 2-QBF$^{\forall}$

2-QBF$^{\forall}$ is the problem of checking the validity of a QBF formula in which only two quantifiers occur, and they are in the order $\forall, \exists$. Specifically, we are interested in the case in which the proposition occurring in such formulas are in CNF.

Definition 51. [2-QBF$^{\forall}$] Let ϕ be a QBF formula of the form $\forall \mathbf{x} \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$, where $\psi(\mathbf{x}, \mathbf{y})$. Is ϕ valid?

Proposition 41. *2-QBF*$^{\forall}$ *is* Π_2^P*-complete.*

We remark that the above proposition holds even in the case $\psi(\mathbf{x}, \mathbf{y})$ is in CNF.

Alphabetical index

CQ, 17
FO, 17
GA, 17
Π_2^p, 13
3-CNF formula, 142

ABox, 14
AC^0, 13
acyclic dependency, 88
answer set, 17
AR-entailment, 90
arity, 11
attribute, 14

basic concept, 14
basic role, 14
BCQ, 16
$BCQ^{\neq}$, 16
Boolean formula, 11
BUCQ, 16
$BUCQ^{\neq}$, 16

C-complete problem, 13
C-hard problem, 13
censor, 22
censor (PPOBDA), 54
censor language, 22
chase, 89
clause, 141
closed formula, 11
closed predicate, 111
closed-head, 113
CNF, 141
comparison atom, 11
concept, 14
conjunctive normal form, 141
coNP, 13
consistent FO theory, 12
constant, 11
cooperativity, 75
CQ, 16
CQ censor, 24
CQ-FO-rewritable, 91
$CQ^{\neq}$, 16
CQE specification, 21

data complexity, 13
database instance, 18
DD censor, 40
decision problem, 13
$DED^{\neq}$, 87
denial assertion, 16
dependency, 87, 88
Description Logics, 13
disjunctive embedded dependency, 87
DL, 13
DL-Lite, 14
DL-Lite$_{\mathcal{R}}$, 14
domain of objects, 48
domain, interpretation, 12

entailment, 12
existential position, 113
extended denial assertion, 32
extended policy, 32

fact, 11
FO rewritable, 17
free variable, 11
frontier variable, 113
functionality, 47

GA censor, 24
Global improvement, 38

Globally-optimal censor, 38
ground, 11
ground atom, 11
guarded dependency, 89

Horn clause, 143
Horn rule, 143

IAR-entailment, 90
IG censor, 39
IGA censor, 28
image, 16
indistinguishability, 24
individual, 14
instantiation, 16
interpretation function, 12
interpretation, FO, 12
inverse property, 14
IP censor, 39

k-DD censor, 41

labelled nulls, 89
linear dependency, 88
literal, 141

mapping assertion, 18
MGU, 12
model, FO, 12
most general unifier, 12

non-redundant denial, 58
non-redundant representation, 32
NP, 13
number restriction, 50
numerical restriction axiom, 50

OBDA, 17
OBDA setting, 19
OBDA specification, 18
OC-AR-entailment, 112
OC-entailment, 112
OC-IAR-entailment, 112
OC-model, 111
OC-repair, 112
OCS, 113
ontology, 14
Ontology-Based Data Access, 17
open predicate, 111
open-closed-separable, 113
open-head, 113
optimal censor, 23

P, 13
Pareto improvement, 38
Pareto-optimal censor, 38
perfect reformulation, 17
policy, 21
PPOBDA specification, 53
predicate, 11
predicate atom, 11
prioritized CQE specification, 37
propositional formula, 141
protection state, 72
PTIME, 13

QBF, 142
quantified Boolean formula, 142
query, 16

recoverability, 91
redundancy condition, 59
repair, 90
retrieved ABox, 18
role, 14

safe $CQ^{\neq}$, 16
safe $UCQ^{\neq}$, 16
secret, 31
sentence, 11
signature, 11
skeptical entailment, 27
source schema, 18
state, 72
sticky dependency, 89
substitution, 12

TBox, 14
term, 11
TGD, 88
theory, FO, 12
topological order, 88
tuple-generating dependency, 88
typing, 47

UCQ, 16
$UCQ^{\neq}$, 16

UNA, 12
unifiable, 12
unifier, 12
unique name assumption, 12
unqualified existential restrictions, 14

valid QBF, 142
value, 14
variable, 11