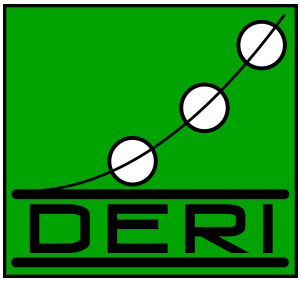
DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE



REDUNDANCY ELIMINATION ON RDF GRAPHS IN THE PRESENCE OF RULES, CONSTRAINTS, AND QUERIES

Reinhard Pichler Axel Polleres
Sebastian Skritek Stefan Woltran

This is a revised version of the original report from April 23, published on 2010 June 7. A preliminary version oc this report has been presented in the 4th Alberto Mendelzon Workshop on Foundations of Data Management, May 2010.

DERI TECHNICAL REPORT 2010-04-23

APRIL 2010; JUNE 2010

DERI Galway
IDA Business Park
Galway, Ireland
www.deri.ie

DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE

DERI TECHNICAL REPORT

DERI TECHNICAL REPORT 2010-04-23, APRIL 2010; JUNE 2010

REDUNDANCY ELIMINATION ON RDF GRAPHS IN THE PRESENCE OF RULES, CONSTRAINTS, AND QUERIES

Reinhard Pichler[1] Axel Polleres[2] Sebastian Skritek[1] Stefan Woltran[1]

Abstract. Based on practical observations on rule-based inference on RDF data, we study the problem of redundancy elimination on RDF graphs in the presence of rules (in the form of Datalog rules) and constraints, (in the form of so-called tuple-generating dependencies), and with respect to queries (ranging from conjunctive queries up to more complex ones, particularly covering features of SPARQL, such as union, negation, or filters). To this end, we investigate the influence of several problem parameters (like restrictions on the size of the rules, the constraints, and/or the queries) on the complexity of detecting redundancy. The main result of this paper is a fine-grained complexity analysis of both graph and rule minimisation in various settings.

Keywords: RDF, Rules, Constraints

---

[1]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria
E-mail: {pichler, skritek, woltran}@dbai.tuwien.ac.at

[2]Digital Enterprise Research Institute, National University of Ireland, Galway
E-mail: axel.polleres@deri.org

Acknowledgements: R. Pichler, S. Skritek and S. Woltran were supported by the Vienna Science and Technology Fund (WWTF), project ICT08-032. A. Polleres was supported by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

This is a revised version of the original report from April 23, published on 2010 June 7. A preliminary version of this report has been presented in the 4th Alberto Mendelzon Workshop on Foundations of Data Management, May 2010.

1 Introduction

The Semantic Web promises to enable computers to gather machine readable meta-data in the form of RDF statements published on the Web and make inferences about these statements by means of accompanying standards such as RDFS and OWL2. While complete OWL2 reasoning is hard – and in many cases even inappropriate for Web data [1] – (incomplete) rule-based inference is becoming quite popular and supported by many RDF Stores and query engines: frameworks like GiaBATA [2], Jena, Sesame, OWLIM,[1] etc. allow for custom inference on top of RDF Stores, supporting different rule-based fragments of RDFS and OWL. Several such fragments have been defined in the literature, such as ρDF [3], DLP [4], OWL$^-$ [5], ter Horst's pD* [6], or SAOR [7], and – more recently – the W3C standardised OWL2RL, a fragment of OWL implementable purely in terms of rule-based inference [8]. All these fragments have in common that they are implementable by simple Datalog-like rules over RDF. As an example, let us take (1) the sub-property rule from RDFS [9, Section 7.3, rule rdfs7], rules (2)–(5) from OWL2RL [9, Section 4.3, rules prp-inv1,prp-symp,prp-spo2] representing inverse properties, symmetric properties, and property chains:[2]

(1) { S P O . P *subPropertyOf* Q . uri(Q) } ⇒ { S Q O }
(2) { S P O . P *inverseOf* Q . uri(O) ∧ uri(Q) } ⇒ { O Q S }
(3) { S P O . P *inverseOf* Q . blank(O) ∧ uri(Q) } ⇒ { O Q S }
(4) { S P O . P *type* *SymmetricProperty* . uri(O) } ⇒ { O P S }
(5) { S P O . P *type* *SymmetricProperty* . blank(O) } ⇒ { O P S }
(6) { S P_0 O_1. ... O_n P_n O. P *propertyChainAxiom* (P_0 ...P_n) } ⇒ { S P O }

Let G_D be an RDF graph talking about authors and their publications:

(7) G_D = { <http://semanticweb.org/wiki/Pat_Hayes> *made* <http://www.w3.org/TR/rdf-mt/>.
(8) <http://semanticweb.org/wiki/Pat_Hayes> *name* "Patrick J. Hayes".
(9) <http://www.w3.org/TR/rdf-mt/> *creator* "Patrick J. Hayes".}

Moreover, let graph G_O be part of the ontology defining the terms used in G_D:

(10) G_O = { *name subPropertyOf label*.
(11) *inverseOf type SymmetricProperty*.
(12) *made inverseOf maker*.
(13) *maker inverseOf made*.
(14) *creator propertyChainAxiom* (*maker label*). }

When storing the graph $G = G_D \cup G_O$ in an RDF Store that supports inference over rules (1)–(6), different questions of redundancy arise like if some statements may be deleted since they can be inferred by the rules. In our example, e.g. statement (9) as well as statement (13) may be deleted, since they could be reproduced by inference. Similarly, suppose that we transfer the graph $G = G_D \cup G_O$ to a "weaker" RDF Store that only supports rules (1)–(3). Then the question is if we thus loose any inferences. In fact, the answer is no. Interestingly enough, standard rule sets, such as OWL2RL are even known to be non-minimal [8, Section 4.3].

We thus want to be able to answer the general question about redundancy of both triples and rules. However, it is often important to limit the minimisation of RDF graphs in such a way that certain consistency

[1] cf. http://jena.sourceforge.net/, http://openrdf.org/, and http://ontotext.com/owlim/

[2] We disregard full URIs for common RDF terms, i.e., we just write e.g. *inverseOf*, for <http://www.w3.org/2002/07/owl#inverseOf>, *name* for <http://xmlns.com/foaf/0.1/name>, or *creator* for <http://purl.org/dc/elements/1.1/>, etc. Further, ($P_1 \dots P_n$) in RDF is short for a fresh variable X plus additional triples X *first* P_1 . X_1 *rest* X_2. ...X_n *first* P_n . X_n *rest* *nil* . using reserved terms *first*, *rest*, *nil*.

conditions must be preserved. These consistency conditions can be expressed by means of constraints [10]. We shall restrict ourselves here to constraints in the form of so-called *tuple-generating dependency (tgd) constraints*, which are a generalisation of the familiar foreign-key dependencies in the relational database world. Roughly speaking, a tgd may be viewed as a generalised rule "read" as constraint. So, for instance, if we read rules (4)-(5) as constraints, we could say that graph G alone without rules satisfies these constraints, and likewise the closure of G with respect to rules (1)-(3) does. Tgd constraints can be more general than (Horn) rules in that they also allow otherwise unbound, existential variables in the head, possibly occurring in a larger conjunct. That is, tgds are – rather than rules – constraining queries (in the head) "triggered" by bindings coming from a query in the body; for instance, a constraint

(15) $\{$ A $made$ D $\} \Rightarrow \{$ A $label$ N . D $creator$ N$\}$

would hold only on graphs where everybody who made something also has a declared label and that label is also used to denote the creator. Note that constraint (15) holds on the closure of G with respect to rule (1) but – as opposed to the constraint reading of (4)-(5) – not on G alone.

Next, we are interested in redundancy with respect to queries. This might be particularly relevant for RDF stores that expose a narrow SPARQL query interface. For instance, suppose that, in our example, we are interested only in completeness with respect to the query "`SELECT ?D ?L` { `?D` $maker$ `?M . ?M` $label$ `?L` }" which is the SPARQL way of writing a conjunctive query:

(16) $\{$ D $maker$ M . M $label$ L $\} \rightarrow ans(D, L)$

In such setting, both rules (3)–(6), as well as triples (9),(11),(13), and (14) can be dropped. Such redundancy elimination is not unique; for instance, keeping triples (11), (13), and rule (4) we could drop (12), still preserving completeness.

The primary goal of our work is a systematic complexity analysis of both graph and rule minimisation under constraints, as well as with respect to queries. To this end, we investigate the influence of several problem parameters (like restrictions on the size of the rules, constraints, and queries) on the complexity of detecting redundancy. A first important step in this investigation has been recently made by Meier [11]. He studied the following problem: Given a graph G, a set $\mathcal{R}$ of rules and a set $\mathcal{C}$ of tgds, can G be reduced to a proper subgraph $G' \subset G$, such that G' still satisfies $\mathcal{C}$ and the closure of G' under $\mathcal{R}$ coincides with the closure of G under $\mathcal{R}$? For the special case that both the rules in $\mathcal{R}$ and the constraints in $\mathcal{C}$ have bounded size (referred to as *b-boundedness*), this problem was shown to be NP-complete in [11]. In this paper, we want to extend the work initiated in [11] and provide a much more fine-grained analysis of the complexity, e.g., by weakening or strengthening restrictions such as b-boundedness and by considering redundancy elimination that only preserves RDF *entailment* (rather than keeping the closure of the original graph under the original rules unchanged) and additionally considering redundancy with respect to queries.

We shall come up with a collection of complexity results, ranging from tractability to Σ_3^P-completeness. Additionally, we address the orthogonal problems of rule minimisation and the problem of reducing rules or triples without preserving completeness of the entire closure, but only ensuring that the answers to certain queries are preserved.

We shall also discuss further variations of the graph and rule minimisation problem. For instance, the rules and tgds in [11] do not allow variables in predicate positions, which is a severe restriction in the sense that many of the common RDF inferences rules are not covered (e.g., all except rules (4) and (5) above). We will not make this restriction, since it can be dropped without significant change of the complexity results.

Organisation of the paper and summary of results. In Section 2, we recall some basic notions and results. A conclusion and an outlook to future work are given in Section 7. Sections 3–6 contain the main results of the paper, namely:

• *Graph Minimisation.* In Section 3, we provide a comprehensive complexity analysis of the RDF graph minimisation problem, both when full reconstruction of the graph or only RDF entailment is required. We study various settings which result from different restrictions on the rules and/or tgds like restricting their size, considering them as fixed, omitting them, or imposing no restrictions at all. Our complexity results range from tractability to Σ_3^P-completeness.

• *Rule Minimisation.* In Section 4, we consider the problem of minimising the set of rules. We show that the problem of finding redundant rules with respect to a given RDF graph is NP-complete for b-bounded rules and not harder than Δ_2^P for arbitrary rules. Note that rule minimisation is closely related to the field of Datalog equivalence and optimisation. We therefore discuss how the large body of results in this area can be fruitfully applied to the problems studied here.

• *Graph Minimisation w.r.t. Queries.* In Section 5, we study how guaranteeing completeness only w.r.t. a given set of conjunctive queries (CQs) or unions of conjunctive queries (UCQs) influences the complexity for each of the above settings. Considering different restrictions on the size of the queries, hardness never exeeds Σ_3^P, but for some settings raises by two levels in the polynomial hierarchy compared to Section 3. Finally we extend our findings to the problem of rule minimisation. We shall also briefly touch on full SPARQL queries beyond unions of conjunctive queries.

• *Problem Variations.* In Section 6, we analyse the complexity of further problems which are either variations of or strongly related to the graph and rule minimisation problems mentioned above. For instance, rather than asking if an RDF graph contains redundant tuples, we consider the problem whether an RDF graph can be reduced below a certain size. We show that this problem is NP-complete also in those settings where the graph minimisation problem is tractable. We also discuss the effect of allowing blank nodes in predicate positions in the Datalog rules.

Due to lack of space, proofs are only sketched. While for most of the hardness proofs we only describe the idea of the reduction, membership proofs are either also informal or even omitted. All proofs are worked out in detail in the appendix.

2 Preliminaries

Let U, B, and L denote pairwise disjoint alphabets for *URI references*, *Blank nodes* (or variables) and *Literals*, respectively. We denote unions of these sets simply by concatenating their names.[3] An RDF statement (or *triple*) is a statement of the form $(s, p, o) \in UB \times U \times UBL$, and an RDF *graph* is a set of triples. In this paper, we do not distinguish between variables and blank nodes, but just note that blank nodes/variables appearing in the data are understood to be existentially quantified within the scope of the whole RDF graph they appear in. We write elements from B (U) as alphanumeric strings starting with an upper case letter (lower case letter or number), elements from L as quoted strings, and – inspired by the common Turtle [13] syntax – RDF statements as white-space separated triples and RDF graphs as '.' separated lists of triples in curly braces.

It is convenient to define the notion of *entailment* between two RDF graphs via the interpolation lemma from [9, Section 2] rather than in a model-theoretic way: an RDF graph G_1 *entails* G_2, written $G_1 \models G_2$ if a subgraph of G_1 is an instance of G_2, that is, if there exists a graph *homomorphism*, i.e., a blank node mapping $\mu : B \to UBL$ such that $\mu(G_2) \subseteq G_1$, where $\mu(G)$ denotes the graph obtained by replacing every variable $\mathtt{B} \in B$ with $\mu(\mathtt{B})$. A homomorphism h' is an *extension* of a homomorphism h if $h'(\mathtt{B}) = h(\mathtt{B})$ for

[3] In this paper, we use a slightly simplified notion of RDF compared to [9], e.g. not considering typed literals separately.

all B on which h is defined. Given G_1, G_2, deciding whether there exists a homomorphism $G_2 \rightarrow G_1$ (thus also $G_1 \models G_2$) is well known to be NP-complete.

We define a *basic graph pattern* (BGP) as a set of generalised triples $(s', p', o') \in UBL \times UBL \times UBL$, a *filter condition* as a conjunct of the unary predicates uri(·), blank(·), literal(·) (denoting the unary relations U, B, and L, respectively). A *filtered basic graph pattern* (FBGP) is a BGP conjoined with a filter condition, the latter containing only variables already appearing in the BGP. Given an FBGP P, we write $BGP(P)$ and $F(P)$ to denote its components, i.e. its BGP and its filter condition, respectively.

We define an *RDF tuple-generating dependency (tgd) constraint* (or simply constraint) r as $\mathcal{A}nte \Rightarrow \mathcal{C}on$, where the *antecedent* $\mathcal{A}nte$ is an FBGP and the *consequent* $\mathcal{C}on$ is a BGP. A constraint $\mathcal{A}nte \Rightarrow \mathcal{C}on$ is a short-hand notation for the first-order formula $\forall\vec{x}\big(\mathcal{A}nte(\vec{x}) \rightarrow (\exists\vec{y})\mathcal{C}on(\vec{x},\vec{y})\big)$ (where $\vec{y}$ denotes the blank nodes occurring in $\mathcal{C}on$ only, while $\vec{x}$ are the remaining blank nodes) Hence, a constraint $\mathcal{A}nte \Rightarrow \mathcal{C}on$ is satisfied over an RDF graph G if for each homomorphism on $\vec{x}$ mapping BGP($\mathcal{A}nte$) to G, there exists an extension h' of h to $\vec{y}$ s.t. $h'(\mathcal{C}on) \subseteq G$. To increase the readability, we will sometimes explicitly write out the quantifiers and variable vectors. *RDF rules* (or simply rules), are syntactically restricted constraints, where all variables appearing in $\mathcal{C}on$ also appear in $\mathcal{A}nte$ (akin to the common notion of safety [14] in Datalog). In the following, we will call RDF rules with an empty filter condition *Datalog rules*. [4] We define the closure of a graph G with respect to a set $\mathcal{R}$ of rules, written $Cl_{\mathcal{R}}(G)$ as usual by the least fix-point of the immediate consequence operator. For a given graph G or a given set $\mathcal{R}$ of rules, we use X_G,$X_{\mathcal{R}}$ ($X \in \{U, B, L\}$) to denote the subset of U (resp. B, L) used in G, or $\mathcal{R}$, respectively.

A *conjunctive query (CQ)* over an RDF graph G is of the form $G_q \rightarrow ans(\tilde{X})$, where G_q is an FBGP, ans is a distinguished predicate, and $\vec{X}$ is a vector of blank nodes. We refer to G_q as the *body* of q ($body(q)$), and to $ans(\tilde{X})$ as the *head* of q ($head(q)$). A *union of conjunctive queries (UCQs)* is a set of CQs, all having the same head. The result of a CQ q over some RDF graph G is defined as the set $q(G) = \{(\vec{x}) \mid$ for all $x_i \in \vec{x}$: $x_i \in U_G B_G L_G U_q L_q$, there exists a homomorphism τ: $B_q \rightarrow U_G B_G L_G$ s.t. $\tau(body(q)) \subseteq G$ and $\tilde{x} = \tau(\tilde{X})\}$. The result of a UCQ is the union of the results of its CQs.

We say that a rule or constraint is *b-bounded* if both, antecedent and consequent contain at most b triples. We say a conjunctive query q is *body-b-bounded* if $body(q)$ is b-bounded, and we denote q as *head-b-bounded* if $|\vec{X}| \leq b$ for some constant b (however, $body(q)$ may be arbitrary). A set $\mathcal{Q}$ of (U)CQs is body-b-bounded (resp. head-b-bounded) if every $q \in \mathcal{Q}$ is body-b-bounded (resp. head b-bounded). Finally, we write $[n]$ to denote the set $\{1, \ldots, n\}$.

3 RDF Graph Minimisation

In this section, we study the complexity of RDF graph minimisation. For different restrictions on the input parameters, the complexity varies between tractability and Σ_3^P-completeness. Formally, we consider the following two basic problems:

Definition 3.1. *Let* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *be the following decision problem:*
INPUT: RDF graph G, set $\mathcal{R}$ of RDF rules, set $\mathcal{C}$ of tgds (G satisfies $\mathcal{C}$).
QUESTION: Is there a $G' \subset G$ s.t. $Cl_{\mathcal{R}}(G') \models Cl_{\mathcal{R}}(G)$ and G' satisfies $\mathcal{C}$?

[4] In fact, we will for most parts of the paper only consider Datalog rules, but will revisit the extension to arbitrary RDF rules in the end of Section 6, concluding that this extension does not change any of our results.

		MINI-RDF$^{\models}$	MINI-RDF$^{\subseteq}$
(1)	$\mathcal{R}$ arb., $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete
(2)	$\mathcal{R}$ arb., $\mathcal{C}$ bb	NP-complete	NP-complete
(3)	$\mathcal{R}$ arb., $\mathcal{C}$ fixed	NP-complete	NP-complete
(4)	$\mathcal{R}$ arb., $\mathcal{C} = \emptyset$	NP-complete	NP-complete
(5)	$\mathcal{R}$ bb., $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete
(6)	$\mathcal{R}$ bb, $\mathcal{C}$ bb	NP-complete	NP-complete [11]
(7)	$\mathcal{R}$ bb, $\mathcal{C}$ fixed	NP-complete	NP-complete
(8)	$\mathcal{R}$ bb, $\mathcal{C} = \emptyset$	NP-complete	in P
(9)	$\mathcal{R}$ fixed, $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete
(10)	$\mathcal{R}$ fixed, $\mathcal{C}$ bb	NP-complete	NP-complete
(11)	$\mathcal{R}$ fixed, $\mathcal{C}$ fixed	NP-complete	NP-complete
(12)	$\mathcal{R}$ fixed, $\mathcal{C} = \emptyset$	NP-complete	in P

Table 1: The complexity of MINI-RDF$^{\models}$ and MINI-RDF$^{\subseteq}$ w.r.t. input parameters ("bb" indicates the set to be b-bounded, and "arb." allows for arbitrary sets.)

Definition 3.2. *Let* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ *be the following decision problem [11]:*
INPUT: RDF graph G, set $\mathcal{R}$ of RDF rules, set $\mathcal{C}$ of tgds (G satisfies $\mathcal{C}$).
QUESTION: Is there a $G' \subset G$ s.t. $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ and G' satisfies $\mathcal{C}$?

The MINI-RDF$^{\subseteq}$ problem and the minimisation of RDF graphs via entailment aim at two kinds of redundancy elimination: In MINI-RDF$^{\subseteq}$, triples which can be restored via the rules are considered as redundant while minimisation via entailment allows us to replace a graph G by $\bar{G} \subset G$ if $\bar{G} \models G$ holds, i.e. checks if G is lean (see [15]). The MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ problem combines these two approaches and thus yields the strongest redundancy criterion. Nevertheless, in most cases, its complexity is not higher than for MINI-RDF$^{\subseteq}$ (see Theorem 3.2).

It is easy to see that the condition $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ in Definition 3.2 is equivalent to $G \subseteq Cl_{\mathcal{R}}(G')$. The following lemma shows that similarly, for MINI-RDF$^{\models}$, it is enough to show $Cl_{\mathcal{R}}(G') \models G$ rather than $Cl_{\mathcal{R}}(G') \models Cl_{\mathcal{R}}(G)$.

Lemma 3.1. *Let G_1, G_2 be RDF graphs and $\mathcal{R}$ a set of rules. Then the following equivalence holds:* $Cl_{\mathcal{R}}(G_2) \models Cl_{\mathcal{R}}(G_1) \Leftrightarrow Cl_{\mathcal{R}}(G_2) \models G_1$.

Theorem 3.2. *For* MINI-RDF$^{\models}$ *and* MINI-RDF$^{\subseteq}$*, the complexity w.r.t. different assumptions on the input (arbitrary, b-bounded, or fixed rule set; arbitrary, b-bounded, fixed, or no constraints) is as depicted in Table 1.*

The following lemma justifies that we do not have to give an explicit completeness proof for each entry in Table 1, and points out a proof plan for Theorem 3.2.

Lemma 3.3. *The graph in Figure 1 correctly describes the dependencies between the problems (identified by their line number) in Table 1, i.e.: If there is an arrow from A to B, then B is a special case of A.*

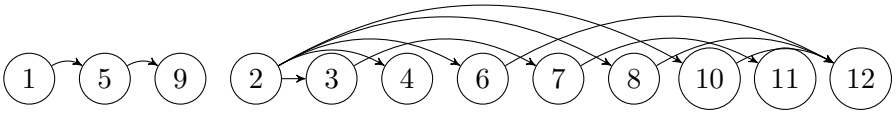


Figure 1: Dependency graph: Numbers refer to lines in Table 1. An arrow from A to B means that B is a special case of A.

Hence an arrow from A to B means that membership results for A hold also for B, and that hardness results for B apply also to A. Therefore, to prove Theorem 3.2, it suffices to show the membership for (1),(2),(8) and the hardness for (4),(9),(11),(12). Due to lack of space, we only work out the hardness results for (9) and (11) (the latter only for MINI-RDF$^{\subseteq}$). Before, we shortly discuss the membership results and give an intuition of why they are correct. All proofs are worked out in detail in the appendix.

The most general case, (1), can be solved by a guess and check algorithm that is allowed to use a Π_2^P oracle for the checks. One has to guess: a subgraph G' of G, a sequence of rule applications on G', and for each rule application a homomorphism justifying that the rule is applicable. Note that $Cl_{\mathcal{R}}(G') \subseteq AD^3$ (with $AD = U_G U_{\mathcal{R}} B_G B_{\mathcal{R}} L_G L_{\mathcal{R}}$). Hence if considering all possible rule applications of length $|AD|^3$, one of them has to return $Cl_{\mathcal{R}}(G')$. The most expensive check is to test if G' satisfies $\mathcal{C}$. However, it obviously fits into Π_2^P.

The following properties lead to the cases of lower complexity: If $\mathcal{R}$ is a b-bounded set, then $Cl_{\mathcal{R}}(G')$ can be computed in polynomial time [11, Proposition 9] and if $\mathcal{C}$ is a b-bounded set, then testing if G' satisfies $\mathcal{C}$ is in PTIME [11, Proposition 3]. For the tractable cases, note that if $\mathcal{C} = \emptyset$, then not all subgraphs of G have to be checked, but only those missing exactly one triple from G.

Lemma 3.4. *The problems* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$*, for fixed $\mathcal{R}$ and arbitrary $\mathcal{C}$, are Σ_3^P-hard.*

Proof. *Σ_3^P-hardness* is shown by reduction from the well-known Σ_3^P-complete problem QSAT$_3$, of which we only give an informal description here. Let an instance of QSAT$_3$ be given by $F = \exists \vec{x_1} \forall \vec{y_1} \exists \vec{x_2} \bigwedge_{i=1}^{n} C_i$, with $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ (clearly, the restriction to 3-CNF is w.l.o.g.). The graph G created contains on the one hand triples encoding truth assignments on clauses (e.g. $\{0\ h_1\ a_{001}\,.\,0\ h_2\ a_{001}\ .\ 1\ h_3\ a_{001}\}$ for the assignment $(false, false, true)$), and on the other hand triples encoding the two possible truth assignments for variables (e.g. $\{v_i\ q_1\ a_{01}\,.\,v_i\ q_1\ a_{10}\}$ for $x_i \in \vec{x_1}$ where v_i is a new URI for each x_i and the URI a_{01} (resp. a_{10}) denotes that x_i evaluates to $false$, hence $\neg x_i$ evaluates to $true$ (resp. x_i to $true$ and $\neg x_i$ to $false$), together with further triples that allow us to actually refer to the truth value of x_i (resp. $\neg x_i$)) under a selected truth assignment. The rules and constraints are chosen in such a way that (1) the triples encoding the truth assignment $(false, false, false)$ for clauses must not be present in any valid subgraph $G' \subset G$, (2) for every $x_i \in \vec{x_1}$ exactly one of the two triples encoding a truth assignment must be present in G' and (3) for all other variables, both triples have to remain in G'. The restrictions imposed by $\bigwedge_{i=1}^{n} C_i$ are encoded in one big tgd, where every homomorphism from its antecedent to G' defines a truth assignment for $\vec{x_1}$ and $\vec{y_1}$. Thereby for every valid G' all such homomorphisms define the same truth assignment on $\vec{x_1}$, hence the values for $\vec{x_1}$ are determined by the selection of G'. But every homomorphism defines a different truth assignment on $\vec{y_1}$, and there exists exactly one homomorphism for each of the $2^{|\vec{y_1}|}$ truth assignments on $\vec{y_1}$. The consequent of the tgd contains a representation of the literals in each clause C_i and has the following property: for every homomorphism h from the antecedent to G', there exists an extension of h to a homomorphism h' from the consequent to G' iff this extension defines a truth assignment on $\vec{x_2}$ such that the assignment on $\vec{x_1}$, $\vec{y_1}$ and $\vec{x_2}$ maps the representations of the clauses onto the possible truth assignments for clauses present in G'. As all triples encoding these truth assignments must be in G', except the ones for $(false, false, false)$ which must not, such an extension for every homomorphism from the antecedent to G' implies that F is valid. □

Lemma 3.5. *The problems* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$*, where both, $\mathcal{R}$ and $\mathcal{C}$ are considered to be fixed, are* NP*-hard.*

Proof. As NP-hardness of MINI-RDF$^{\models}$ follows easily from the coNP-hardness of testing if G is lean [15], we concentrate on MINI-RDF$^{\subseteq}$ and prove its NP-hardness by reduction from the 3-SAT problem. We fix

the rules and tgds as

$$\begin{aligned}\mathcal{R} &= \{ \quad \{X' \; in \; I \,.\, X \; active \; I\} \Rightarrow \{X' \; active \; I\}\}\\ \mathcal{C} &= \{ \quad \{X \; active \; I \,.\, X \; in \; J\} \Rightarrow \{X \; active \; J\}\\ &\qquad \{X \; clash \; X' \,.\, X \; active \; I \,.\, X' \; active \; I' \,.\, Y \; in \; J\} \Rightarrow \{Y \; active \; J\}\}.\end{aligned}$$

Now let an instance of 3-SAT be given by the formula $F = C_1 \wedge \cdots \wedge C_n$, where $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ and the $l_{i,j}$ are literals. W.l.o.g., we assume that every variable appears negated and unnegated in F. Then we construct an RDF graph $G = \{l^*_{i,j} \; in \; c_i \mid i \in [n], j \in [3]\} \cup \{l^*_{i,j} \; active \; c_i \mid i \in [n], j \in [3]\} \cup \{x_j \; clash \; \bar{x}_j \mid x_j \text{ in } F\}$, where we introduce new URIs c_i (for every clause C_i) and $x_j, \bar{x}_j$ (for every variable x_j in F), and $l^*_{i,j} = x_j$ (resp. $\bar{x}_j$) if $l_{i,j} = x_j$ (resp. $\neg x_j$).
Intuitively, the triples in G with predicate in encode the literals in F. If a triple with predicate $active$ remains in the selected subgraph G' then the corresponding literal in F is set to true. The triples with $clash$ keep track of dual literals. □

4 Rule Minimisation

In this section, we study the rule minimisation problem of RDF graphs. Although there is a huge amount of literature in the Datalog world addressing related problems (as query containment), the particular nature of the problems we study, requires a distinguished complexity analysis. Note that rules for RDF, when written as Datalog rules, have a fixed predicate arity of three, which makes problems computationally easier than in the general Datalog setting (see, e.g. [16]). Depending on whether we consider the Datalog rules as b-bounded or not, we obtain complexity results from NP-completeness to Δ_2^P-membership. The rule minimisation problem is formally defined as follows. As the RDF graph remains unchanged, constraints are irrelevant here.

Definition 4.1. *Let* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *be the following decision problem:*
INPUT: An RDF graph G and a set $\mathcal{R}$ of RDF rules.
QUESTION: Does there exist $\mathcal{R}' \subset \mathcal{R}$ s.t. $Cl_{\mathcal{R}'}(G) \models Cl_{\mathcal{R}}(G)$?

Definition 4.2. *Let* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ *be the following decision problem:*
INPUT: An RDF graph G and a set $\mathcal{R}$ of RDF rules.
QUESTION: Does there exist $\mathcal{R}' \subset \mathcal{R}$ s.t. $Cl_{\mathcal{R}'}(G) = Cl_{\mathcal{R}}(G)$?

For the case that the set of rules is b-bounded, we can pinpoint the complexity of the problem to NP.

Theorem 4.1. *For a set $\mathcal{R}$ of b-bounded rules (for fixed b), the problem* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *is* NP*-complete while* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ *is in* PTIME.

Proof. The hardness is shown by reduction from the 3-Colorability problem. The RDF graph G is built over the URIs $U = \{0, 1, 2\}$ in subject and object positions. G contains triples of the form $i \; e \; j$ for all value combinations $i, j \in U$ with $i \neq j$. $\mathcal{R}$ contains a single rule which generates an encoding $X_\alpha \; e \; X_\beta$ (with blank nodes X_α, X_β) for each edge (v_α, v_β) of the graph to be 3-colored. This rule is redundant iff a valid 3-coloring exists, i.e., iff the triples $X_\alpha \; e \; X_\beta$ can be mapped into $\{i \; e \; j \mid i \neq j\}$.

For the membership, note that it suffices to compare the closure of G under $\mathcal{R}$ with the closure of G under every subset of $\mathcal{R}$ missing exactly one rule. In the b-bounded case, the closure can be computed efficiently. Hence, we get PTIME-membership for RDF-RULEMIN$^{\subseteq}$ and NP-membership for RDF-RULEMIN$^{\models}$ (the NP-computation is needed only for the entailment check). □

Theorem 4.2. *For arbitrary rules,* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ *is* coNP*-hard and in* Δ_2^P *while* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *is* NP*-hard,* coNP*-hard, and in* Δ_2^P.

Proof. The Δ_2^P upper bound is due to the fact that computing the closure under a set of arbitrary rules requires an NP-oracle (to check if a rule is applicable). The NP-hardness of RDF-RULEMIN$^{\models}(G, \mathcal{R})$ carries over from Theorem 4.1. The coNP-hardness of both problems is shown by a straightforward reduction from the co-problem of 3-Colorability: $\mathcal{R}$ contains a single rule whose body encodes the graph to be 3-colored. This rule is redundant iff no 3-coloring exists. □

In order to reduce the complexity of the problems RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ and RDF-RULEMIN$^{\models}(G, \mathcal{R})$, one could seek for approximations of those problems. In fact, one option is to check for *redundant* rules in the set $\mathcal{R}$ of given Datalog rules; or whether some rule is subsumed by another rule from $\mathcal{R}$. The first problem is known to be tractable while the test for rule subsumption is NP-complete (see [17]). The latter result can be shown to hold also for rules of bounded arity (which we deal with here); but becomes tractable in the case of b-bounded rules. Further methods (e.g., folding and unfolding of rules) are well understood for logic programs (see [18]), and could also apply to our domain. An in-depth analysis how to use those results in our setting is left for future work.

5 Minimisation w.r.t. Queries

Another variant of the RDF graph and rule minimisation problems is to guarantee completeness only w.r.t. a given set of queries. We restrict ourselves here to (unions of) conjunctive queries (CQs resp. UCQs). Such a minimisation is of high interest, e.g. when importing data into an RDF Store that provides a narrow query interface only. Formally, we get the following problems:

Definition 5.1. MINI-RDF$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$ *is the following decision problem:*
INPUT: An RDF graph G*, a set* $\mathcal{R}$ *of RDF rules, a set* $\mathcal{C}$ *of tgds (*G *satisfies* $\mathcal{C}$*), and a set* $\mathcal{Q}$ *of CQs.*
QUESTION: Is there a $G' \subset G$ *s.t. (1) for every* $q \in \mathcal{Q}$*, the answers to* q *over* $Cl_{\mathcal{R}}(G)$ *coincide with the answers to* q *over* $Cl_{\mathcal{R}}(G')$ *and (2)* G' *satisfies* $\mathcal{C}$*?*

Definition 5.2. RDF-RULEMIN$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{Q})$ *is the following decision problem:*
INPUT: An RDF graph G*, a set* $\mathcal{R}$ *of RDF rules, and a set* $\mathcal{Q}$ *of CQs.*
QUESTION: Is there a $\mathcal{R}' \subset \mathcal{R}$ *s.t. for every* $q \in \mathcal{Q}$*, the answers to* q *over* $Cl_{\mathcal{R}}(G)$ *coincide with the answers to* q *over* $Cl_{\mathcal{R}'}(G)$*?*

First, we observe that all hardness results from Sections 3 and 4 carry over to the CQ-variants. To see this, we note that, for instance, MINI-RDF$^{\subseteq}$ corresponds to the special case of MINI-RDF$^{\subseteq, CQ}$ where $\mathcal{Q} = \{\{S\ P\ O\} \rightarrow ans(S, P, O)\}$.

Analogously to the various settings studied in the previous sections resulting from different restrictions on $\mathcal{C}$ and $\mathcal{R}$, we also study three settings of the CQ-variants of these problems by considering $\mathcal{Q}$ to be body-b-bounded, head-b-bounded, or unrestricted, respectively. We thus get the following complexity results.

Theorem 5.1. *For* MINI-RDF$^{\subseteq, CQ}$*, the complexity w.r.t. different assumptions on the input (arbitrary, b-bounded or fixed rule set; arbitrary, b-bounded, fixed, or no constraints; body-b-bounded, head b-bounded, or arbitrary CQs) is as depicted in Table 2, rows (1) – (12). Likewise, the complexity of* RDF-RULEMIN$^{\subseteq, CQ}$ *is depicted in Table 2, rows (I) – (II).*

		$\mathcal{Q}$ body-bb (a)	$\mathcal{Q}$ head-bb (b)	$\mathcal{Q}$ arb. (c)
(1)	$\mathcal{R}$ arb., $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete	Σ_3^P-complete
(2)	$\mathcal{R}$ arb., $\mathcal{C}$ bb	NP/ Δ_2^P	NP/ Δ_2^P	Σ_3^P-complete
(3)	$\mathcal{R}$ arb., $\mathcal{C}$ fixed	NP/ Δ_2^P	NP/ Δ_2^P	Σ_3^P-complete
(4)	$\mathcal{R}$ arb., $\mathcal{C} = \emptyset$	NP/ Δ_2^P	NP/ Δ_2^P	Π_2^P-complete
(5)	$\mathcal{R}$ bb., $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete	Σ_3^P-complete
(6)	$\mathcal{R}$ bb, $\mathcal{C}$ bb	NP-complete	NP/ Δ_2^P	Σ_3^P-complete
(7)	$\mathcal{R}$ bb, $\mathcal{C}$ fixed	NP-complete	NP/ Δ_2^P	Σ_3^P-complete
(8)	$\mathcal{R}$ bb, $\mathcal{C} = \emptyset$	in P	NP/ Δ_2^P	Π_2^P-complete
(9)	$\mathcal{R}$ fixed, $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete	Σ_3^P-complete
(10)	$\mathcal{R}$ fixed, $\mathcal{C}$ bb	NP-complete	NP/ Δ_2^P	Σ_3^P-complete
(11)	$\mathcal{R}$ fixed, $\mathcal{C}$ fixed	NP-complete	NP/ Δ_2^P	Σ_3^P-complete
(12)	$\mathcal{R}$ fixed, $\mathcal{C} = \emptyset$	in P	NP/ Δ_2^P	Π_2^P-complete
(I.)	$\mathcal{R}$ arb.	coNP/ Δ_2^P	coNP + NP/ Δ_2^P	Π_2^P-complete
(II.)	$\mathcal{R}$ bb.	in P	NP/ Δ_2^P	Π_2^P-complete

Table 2: The complexity of MINI-RDF$^{\subseteq,CQ}$ (1-12) and RDF-RULEMIN$^{\subseteq,CQ}$ (I. - II.) w.r.t. input parameters ("bb" stands for "b-bounded", and "arb." for "arbitrary").

Thereby (co-)NP / Δ_2^P *denotes the lower bound / upper bound for the complexity. We write* coNP + NP/ Δ_2^P *if both,* coNP- *and* NP- *hardness hold. All lower bounds hold even if Q consists of a single CQ. Likewise, all upper bounds hold even if $\mathcal{Q}$ is a set of UCQs.*

Obviously, body-b-bounded (U)CQs are a special case of head-b-bounded (U)CQs, which in turn are a special case of arbitrary (U)CQs. By combining this observation with Lemma 3.3, to prove Theorem 5.1, it suffices to show membership for the entries (6a), (8a), (2b), (1c), (4c) as well as (Ia), (IIb), and (IIc) in Table 2, and hardness for (11a), (12b), (11c), (12c) as well as (IIa), (Ib), and (Ic). Due to space restrictions, we only give a rough sketch of the intuition of these results. All proofs are worked out in detail in the appendix.

Membership of the most general case (1c) is shown by considering the following algorithm: guess a subset $G' \subset G$ and check with Π_2^P-oracles if G' satisfies $\mathcal{C}$ and if $q(\hat{G}) = q(\hat{G}')$, where $\hat{G} = Cl_{\mathcal{R}}(G)$, resp. $\hat{G}' = Cl_{\mathcal{R}}(G')$. Moreover, the closures $\hat{G}$ and $\hat{G}'$ can be computed in Δ_2^P, since they are subsets of AD^3.

The other columns contain potentially easier settings because of the restrictions on the queries, while the other rows are potentially easier because of restrictions on $\mathcal{R}$ and $\mathcal{C}$. In particular, if no constraints are present, it suffices to check the "direct" subsets $G' = G \setminus \{t\}$ for each $t \in G$. Thus the non-deterministic guess of $G' \subset G$ is no longer needed. By the same token, rule minimisation is not harder than Π_2^P, since we only need to check the direct subsets $\mathcal{R}' = \mathcal{R} \setminus \{r\}$. If the queries are head-b-bounded, then there are at most polynomially many candidates for answer-tuples. Hence, to answer a query q over two different RDF graphs is feasible in Δ_2^P (rather than Π_2^P). For body-b-bounded queries, the answers to a query q over an RDF graph can even be computed in PTIME.

Turning to the lower bounds, the NP-hardness for (11a) follows immediately from the above remark that the hardness results of MINI-RDF$^{\subseteq}$ carry over. (12b) differs from the previous setting by allowing more expressive queries, but no constraints (which, for MINI-RDF$^{\subseteq}$, leads to tractability). However, the NP-hardness of this case follows immediately from the coNP-hardness of checking if an RDF graph is lean [15] and defining $\mathcal{Q} = \{G \rightarrow ans()\}$. The hardness for (11c) is shown by reduction from QSAT$_3$. Its main idea is, given a formula $F = \exists \vec{x}_1 \forall \vec{y}_1 \exists \vec{x}_2 \phi$, to define a CQ q and a graph G such that every homomorphism

$\tau\colon body(q) \to G$ defines a truth assignment on the variables in F. (The proof allows even $\mathcal{R} = \emptyset$.) Thereby q outputs the values of this truth assignment on $\vec{y}_1$. G is further chosen in such a way that $q(G)$ contains an encoding of all possible truth assignments on $\vec{y}_1$. The constraints in $\mathcal{C}$ are such that over every proper subgraph $G' \subset G$ that satisfies $\mathcal{C}$, every homomorphism from $body(q)$ to G' now encodes truth assignments that actually satisfy ϕ. At the same time, the assignment on $\vec{x}_1$ is already defined by the choice of G'. Hence, if $q(G')$ also contains encodings for all possible truth assignments on $\vec{y}_1$, this means that F is indeed satisfied. For $\mathcal{C} = \emptyset$, we only get Π_2^P-hardness since we can no longer express that valid choices for G' encode a truth assignment on $\vec{x}_1$.

For the rule minimisation, the Π_2^P-hardness is shown similarly to the Π_2^P-hardness in case (12c). In case of (head-/body-)b-bounded queries, the answers to the queries can no longer produce all possible truth assignments on $\vec{y}_1$. Hence, we can only prove NP- and coNP-hardness, respectively, in cases (Ia) and (IIa).

5.1 Beyond Conjunctive Queries – SPARQL

RDF minimization w.r.t. (unions of) conjunctive queries could be extended to more expressive query languages. Actually, it can be checked that all upper bounds proved in this section are still valid if the CQs are allowed to contain negation in the body. In particular, the complexity of the problems considered here does not go beyond Σ_3^P for this kind of extension (we work out the details in the appendix). In contrast, if we allow arbitrary non-recursive datalog queries with negation (a query language which – as well known – covers all of SPARQL [19]), then the complexity of the problems considered here will be dominated by the complexity of query evaluation, which is PSPACE-complete in this case, see [20]. We a more fine-grained analysis of different fragments of SPARQL to future work.

6 Problem Variations

In this section, we discuss some further problems which are variations of or strongly related to the problems studied in the previous sections. We start by a variation of the graph minimisation problem. But now we ask if G can be replaced by a subgraph G' whose size is bounded by some given bound k (rather than an arbitrary subgraph $G' \subset G$). Formally, we study the following problem.

Definition 6.1. *Let* MINI-RDF$^{card}(G, \mathcal{R}, \mathcal{C}, k)$ *be the following decision problem: INPUT: An RDF graph G, a set $\mathcal{R}$ of RDF rules, a set $\mathcal{C}$ of tgds and integer k.*
QUESTION: Does there exist a subgraph $G' \subset G$ with $|G'| \leq k$, s.t. G' satisfies $\mathcal{C}$ and $G \subseteq Cl_{\mathcal{R}}(G')$?

It can be easily verified that for all cases in Table 1 that are at least NP-hard, the complexity for MINI-RDF^{card} does not change. Intuitively, this is because the nondeterministic algorithms for solving these problems all start with "guess a subgraph $G' \subset G$", which can be easily changed to "guess a subgraph with at most k triples". Therefore, the only two interesting cases are MINI-RDF$^{\subseteq}$ with a b-bounded or fixed set $\mathcal{R}$ and no constraints, as they can be decided in PTIME. We show that for MINI-RDF^{card}, the complexity goes up to NP-completeness.

Theorem 6.1. *The problem* MINI-RDF$^{card}(G, \mathcal{R}, \mathcal{C}, k)$ *is* NP*-complete if $\mathcal{C} = \emptyset$ and $\mathcal{R}$ is either considered as fixed or a set of b-bounded rules (for fixed b).*

Proof. The *hardness* proof is by reduction from the Vertex Cover problem. We give the basic ideas of this reduction. Given some graph $G = (V, E)$, the RDF graph G^{rdf} contains one distinct triple for every $v \in V$.

The intuition is that the subset of those triples contained in a valid subgraph $G' \subset G^{rdf}$ describes a vertex cover. We further have three rules, one that (given $G' \subset G$) adds all edges covered by the remaining vertices in G', one that (by repeated application) checks whether all edges are covered, and finally one rule that, if indeed all edges are covered, allows to restore the vertices from $G^{rdf} \setminus G'$. To allow to express according rules, G^{rdf} contains triples encoding further information (like e.g. neighbourhood of vertices and edges). But as they cannot be derived by any rule, they must remain unchanged in any valid $G' \subset G^{rdf}$. Further, their number (say K) only depends on G, such that there exists a vertex cover of size k iff there exists a valid $G' \subset G^{rdf}$ of size $K + k$. □

Next we want to identify the sources of the complexity of MINI-RDF$^{\models}$ and MINI-RDF$^{\subseteq}$ for the cases where $\mathcal{C}$ is allowed to contain arbitrary tgds. We show that the complexity is independent of the rules, but arises mainly from the question whether there exists some non-empty subgraph that satisfies all constraints.

Theorem 6.2. *Let G be a RDF graph and $\mathcal{C}$ a set of tgds. Deciding whether there exists some $\emptyset \neq G' \subset G$ s.t. G' satisfies $\mathcal{C}$ is Σ_3^P-complete.*

Proof. Membership follows from Theorem 3.2. Hardness is shown by a modification of the reduction given in the proof of Lemma 3.4. We give the intuition of these modifications. In the aforementioned proof, the intuitive meaning of the rules, together with the requirement $G \subseteq Cl_{\mathcal{R}}(G')$, was that for each $v_i \in \vec{x_1}$, either $\{v_i\ q_1\ a_{01}\}$ or $\{v_i\ q_1\ a_{10}\}$ has to remain in the subgraph G'. However, this can be also formulated as a constraint. By introducing an additional triple for every $v_i \in \vec{x_1}$ (e.g. $\{v_i\ opt\ v_i\}$) that is enforced to be contained in any non-empty subgraph, the tgd $\{V\ opt\ V\} \Rightarrow \{V\ q_1\ A\}$ does the job. □

From the (full) proof of Lemma 3.5, it follows that for MINI-RDF$^{\models}$, one source of the NP-hardness is just to decide the entailment. However, similarly to the last theorem, we can show that for b-bounded tgds, just testing for the existence of a valid subgraph already contains the full hardness too.

Theorem 6.3. *Let G be an RDF graph and $\mathcal{C}$ a set of b-bounded tgds. Deciding whether there exists some $\emptyset \neq G' \subset G$ s.t. G' satisfies $\mathcal{C}$ is* NP*-complete.*

Proof. Membership follows from Theorem 3.2. *Hardness* is shown by reduction from the SAT problem. The reduction is very similar to the one of Lemma 3.5, only that all the implicit information about which triples must not be removed from G (expressed by not providing rules to derive them) now have to be made explicit as tgds. This however no longer allows for a fixed set of tgds, but makes the number of tgds dependent on F. □

Recall that tgds generalize (safe) datalog rules by allowing existential quantification and conjunctions in the head. In other words, datalog rules are an important special case of tgds – referred to as *full tgds* in the information integration literature. Below, we show that restricting the constraints to full tgds pushes the Σ_3^P-completeness results from Theorems 3.2 and 6.2 down to Σ_2^P.

Theorem 6.4. *The problems* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ *are Σ_2^P-complete if $\mathcal{C}$ is a set of full tgds. Σ_2^P-completeness even holds for fixed $\mathcal{R}$.*

Likewise, let G be an RDF graph and $\mathcal{C}$ a set of full tgds. Deciding whether there exists some $\emptyset \neq G' \subset G$ s.t. G' satisfies $\mathcal{C}$ is Σ_2^P-complete.

Proof. The Σ_2^P*-membership* is established by the same algorithm as the Σ_3^P-membership in case of unrestricted tgds according to Theorem 3.2. However, by the restriction to full tgds, we now only need a coNP-oracle (rather than Π_2^P) for checking that the tgds are satisfied. The Σ_2^P*-hardness* is shown via reduction from QSAT$_2$ by using similar ideas as in the Σ_3^P-hardness proof in Lemma 3.4. □

6.1 General RDF Rules vs. Datalog Rules

We now show that the complexity of the investigated problems remains unchanged by allowing additional predicates $uri(.)$, $blank(.)$, $lit(.)$ to restrict the type of a value in a Datalog rule, that is, allowing general RDF rules as defined in Section 2. Note that for every $x \in U \cup B \cup L$ occurring in some RDF-graph G (i.e. for every element of the active domain) it can be easily recognised whether it belongs to U, B or L: This could be either decided using syntactic criteria, or by a lookup in U, B and L (although those sets are supposed to be countable infinite, one can assume that U_G, B_G and L_G, i.e. the elements of the active domain, are the "first" elements of these sets). Therefore, determining the type of some element requires at most polynomial time in the size of G. Therefore, for every element x of the active domain of G, we create a ground atom $B_t(x)$, $U_t(x)$ or $L_t(x)$, depending on the type of x. By encoding an atom $blank(X)$ as triple $\{X\ blank\ X\}$ in G, we can make this information available for rule application without increasing the complexity of the problem.

The same argument allows us to overcome the problem that the closure w.r.t. a rule set $\mathcal{R}$ contains invalid RDF triples (containing e.g. a blank node in a predicate position). Depending on whether invalid triples are allowed in intermediate results or not, we can pursue one of the following two strategies: (i) in a post-processing step, we can check for every triple in $\mathcal{R}(G)$ whether it is valid or not. In the latter case, it is removed; (ii) if invalid triples should also be excluded from any intermediate results, then the rules can be (automatically) augmented by at most 2 additional predicates in the rule body, $urib(A)$ and $uri(B)$, assuming that the rule head is $\{A\ B\ C\}$. The predicate $urib(.)$ can be easily defined from $uri(.)$ and $blank(.)$ by e.g. $\{uri(X)\} \Rightarrow \{urib(X)\}$ and $\{blank(X)\} \Rightarrow \{urib(X)\}$. This is similar to variations of rules (2)–(4) in Section 1, where the filter conditions guaranteed valid intermediate triples.

7 Conclusion

We proved a collection of complexity results for minimisation problems over RDF graphs where we considered various restrictions on the rules and tgds. One such restriction was b-boundedness [11]. We note that this restriction can be relaxed by bounding not necessarily the size of the rules (or tgds) but only the maximal number of blank nodes occurring in the rules (or tgds) — in the Datalog world, Vardi [21] showed that such a restriction decreases complexity. We further discussed how the complexity of the problem increases if one requires completeness only with respect to a given set of conjunctive queris (CQs). Notably, if the CQs are restricted to have bounded head arity, while providing additional minimisation potential, the problem becomes only mildly harder.

The minimisation problems considered here are driven by practical needs to represent RDF data compactly or tailor them to engines supporting different rule sets. Our results also provide a basis for eliminating redundancies in existing practically relevant rule sets, such as OWL2RL [8]. We believe that our results will gain even more relevance with the advent of novel standards such as the W3C rule interchange format (RIF) which will allow one to enrich RDFS and OWL with Web-publishable custom rule sets [22].

As future work, our investigations should be further extended in several directions such as a more fine-grained analysis of SPARQL fragments when reduncdancy w.r.t. queries is considered, for instance well-designed SPARQL queries [20]. Moreover, we plan to cast the obtained results into practical algorithms to "compress" RDF graphs and rule sets, investigate related relevant problems such as "trading" triples for rules, or vice versa, and experimentally evaluating effects of such transformations on query answering with dynamic inference such as sketched in [2].

References

[1] Hogan, A., Decker, S.: On the ostensibly silent 'W' in OWL 2 RL. In: Proc. RR'09. Volume 5837 of LNCS., Springer (2009) 118–134

[2] Ianni, G., Krennwallner, T., Martello, A., Polleres, A.: Dynamic querying of mass-storage RDF data with rule-based entailment regimes. In: Proc. ISWC'09. Volume 5823 of LNCS., Springer (2009) 310–327

[3] Muñoz, S., Pérez, J., Gutiérrez, C.: Minimal deductive systems for RDF. In: Proc. ESWC'07. Volume 4519 of LNCS., Springer (2007) 53–67

[4] Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logics. In: Proc. WWW'03. (2003) 48–57

[5] de Bruijn, J., Polleres, A., Lara, R., Fensel, D.: OWL$^-$. WSML D20.1v0.2 (2005)

[6] ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. J. Web Sem. **3**(2-3) (2005) 79–115

[7] Hogan, A., Harth, A., Polleres, A.: Scalable authoritative OWL reasoning for the web. International Journal on Semantic Web and Information Systems **5**(2) (2009)

[8] Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web ontology language profiles (October 2009) W3C Recommendation.

[9] Hayes, P.: RDF semantics. Technical report, W3C (February 2004) W3C Recommendation.

[10] Lausen, G., Meier, M., Schmidt, M.: SPARQLing constraints for RDF. In: Proc. EDBT'08, ACM Press

[11] Meier, M.: Towards Rule-Based Minimization of RDF Graphs under Constraints. In: Proc. RR'08. Volume 5341 of LNCS., Springer (2008) 89–103

[12] Pichler, R., Polleres, A., Skritek, S., Woltran, S.: Minimizing RDF graphs under rules and constraints revisited. Technical report, DERI (April 2010) Available at `http://www.deri.ie/fileadmin/documents/DERI-TR-2010-04-23.pdf`.

[13] Beckett, D., Berners-Lee, T.: Turtle - Terse RDF Triple Language (January 2008) W3C Team Submission, `http://www.w3.org/TeamSubmission/turtle/`.

[14] Ullman, J.D.: Principles of Database and Knowledge Base Systems. Computer Science Press, New York, NY, USA (1989)

[15] Gutierrez, C., Hurtado, C., Mendelzon, A.: Foundations of semantic web databases. In: Proc. 04, ACM (2004) 95–106

[16] Eiter, T., Faber, W., Fink, M., Woltran, S.: Complexity results for answer set programming with bounded predicate arities and implications. Ann. Math. Artif. Intell. **51**(2-4) (2007) 123–165

[17] Eiter, T., Fink, M., Tompits, H., Traxler, P., Woltran, S.: Replacements in non-ground answer-set programming. In: Proc. KR'06, AAAI Press (2006) 340–351

[18] Pettorossi, A., Proietti, M.: Transformation of logic programs. In Gabbay, D.M., Hogger, C.J., Robinson, J.A., eds.: Handbook of Logic in Artificial Intelligence and Logic Programming. Volume 5., Oxford University Press (1998) 697–787

[19] Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: Proc. ISWC 2008. Volume 5318 of LNCS., Springer (2008) 114–129

[20] Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. **34**(3) (2009)

[21] Vardi, M.: On the complexity of bounded-variable queries. In: Proc. PODS'95, ACM Press (1995) 266–276

[22] de Bruijn, J.: RIF RDF and OWL Compatibility (May 2010) W3C Proposed Recommendation, `http://www.w3.org/TR/2010/PR-rif-rdf-owl-20100511/`.

Appendix

A Full proofs for Section 3

Lemma 3.1. *Let G_1, G_2 be RDF graphs and $\mathcal{R}$ a set of rules. Then the following equivalence holds:* $Cl_{\mathcal{R}}(G_2) \models Cl_{\mathcal{R}}(G_1) \Leftrightarrow Cl_{\mathcal{R}}(G_2) \models G_1$.

Proof. The "$\Rightarrow$"-direction is trivial. To show the "$\Leftarrow$"-direction, recall that the closure $Cl_{\mathcal{R}}(G)$ of a graph G under a set of rules $\mathcal{R}$ is defined by the least fixpoint of the immediate consequence operator.

Let $r \in \mathcal{R}$ with $r = \forall \vec{x}\big(\mathcal{A}nte(\vec{x}) \rightarrow \mathcal{C}on(\vec{x})\big)$. A single application of the immediate consequence operator T w.r.t. r extends the graph G_1 to the graph $G_1' = G_1 \cup \mathcal{C}on(\lambda(\vec{x}))$, where λ is a mapping on $\vec{x}$ with $\mathcal{A}nte(\lambda(\vec{x})) \subseteq G_1$. It suffices to prove the implication $Cl_{\mathcal{R}}(G_2) \models G_1 \Rightarrow Cl_{\mathcal{R}}(G_2) \models G_1'$. From this, the lemma follows by induction on the number of rule applications when computing the closure $Cl_{\mathcal{R}}(G_1)$.

By assumption, $\mathcal{A}nte(\lambda(\vec{x})) \subseteq G_1$. Moreover, $Cl_{\mathcal{R}}(G_2) \models G_1$ holds. Hence, there exists a mapping η on the blank nodes in G_1, s.t. $\eta(G_1) \subseteq G_2$. In total, for $\sigma = \eta \circ \lambda$, we thus have $\mathcal{A}nte(\sigma(\vec{x})) \subseteq G_2$. Since $Cl_{\mathcal{R}}(G_2)$ is closed under $\mathcal{R}$, the inclusion $\mathcal{C}on(\sigma(\vec{x}))) \subseteq G_2$ holds as well. We claim that η is the desired homomorphism $\eta\colon G_1' \rightarrow G_2$. Clearly, we have $\eta(G_1) \subseteq G_2$ by the above considerations. It remains to show that η also maps the triples in $G_1' \setminus G_1$ to G_2. Consider $G_1' \setminus G_1 = \mathcal{C}on(\lambda(\vec{x}))$. By definition, $\sigma = \eta \circ \lambda$. Moreover, $\mathcal{C}on(\sigma(\vec{x})) \subseteq G_2$. Hence, $\eta(\mathcal{C}on(\lambda(\vec{x}))) = \mathcal{C}on(\eta \circ \lambda(\vec{x})) = \mathcal{C}on(\sigma(\vec{x})) \subseteq G_2$. In total, we thus have $\eta(G_1') \subseteq G_2$ and, therefore, $Cl_{\mathcal{R}}(G_2) \models G_1'$. □

Lemma 3.3. *The graph in Figure 1 contains an arrow from A to B then B is a special case of A.*

Proof. The Lemma follows immediately from the observation that arbitrary sets $\mathcal{R}$ (resp. $\mathcal{C}$) include b-bounded sets $\mathcal{R}$ (resp. $\mathcal{C}$), which in turn include sets of constant size. Finally, the empty set is of constant size. □

Theorem 3.2. *For* MINI-RDF$^{\models}$ *and* MINI-RDF$^{\subseteq}$*, the complexity w.r.t. different assumptions on the input (arbitrary, b-bounded, or fixed rule set; arbitrary, b-bounded, fixed, or no constraints) is as depicted in Table 1.*

Proof. As mentioned in the main body, to prove the theorem, it suffices to show the membership for (1) (Lemma A.1) (2) (Lemma A.2) and (8) (Lemma A.3) and the hardness for (4) (Lemma A.4) (9) (Lemma 3.4) (11) (Lemma 3.5) (12) (Lemma A.5). □

In the following we give the detailed proofs of the proof sketches in the main body of the text, and formally state and prove the missing results.

Lemma 3.4. *The problems* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$*, for fixed $\mathcal{R}$ and arbitrary $\mathcal{C}$ are Σ_3^P-hard.*

Proof. As already mentioned in the main body of the text, the hardness is shown by reduction from QSAT_3. We will start proving the hardness of MINI-RDF$^{\subseteq}$. However, as the graph G defined in this reduction does not contain any blank nodes, the reduction gives a positive instance of MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ iff it is also a positive instance of MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ (as there are no blank nodes in G, every homomorphism defined on G must be the identity mapping). Hence the same reduction proves the hardness of MINI-RDF$^{\models}$ as well.

Let an arbitrary instance of QSAT$_3$ be given by $F = \exists\vec{x}_1\forall\vec{y}_1\exists\vec{x}_2 \bigwedge_{i=1}^{n} C_i$, with $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ (obviously, the restriction to 3CNF is w.l.o.g.). Let $|\vec{x}_1| = r$, $|\vec{y}_1| = s$ and $|\vec{x}_2| = t$ and let V denote the set $V = \vec{x}_1 \cup \vec{x}_2 \cup \vec{y}_1$ where we write $V = \{v_i \mid i \in [r+s+t]\}$ and $v_i \in \begin{cases} \vec{x}_1 & \text{if } i \leq r \\ \vec{y}_1 & \text{if } i > r \text{ and } i \leq r+s \\ \vec{x}_2 & \text{if } i > r+s \end{cases}$
By slight abuse of notation, we will use v_i to denote both, the variables in F as well as URIs in G, where we assume one unique URI v_i for every variable v_i in F.

To provide a better understanding of the reduction, we first present its basic idea in terms of ternary relations (note that RDF triples correspond to binary relations), and then translate them into RDF triples.

Given F, we define an instance $\bar{G} = \bar{G}_{01} \cup \bar{G}_{000} \cup \bar{G}_{D_1} \cup \bar{G}_{D_2}$ over the relational schema $S = \{C/3, D_1/3, D_2/3\}$ as follows:
$\bar{G}_{000} = \{C(0,0,0)\}$,
$\bar{G}_{01} = \{C(0,0,1), C(0,1,0), C(1,0,0), C(0,1,1), C(1,0,1), C(1,1,0), C(1,1,1)\}$,
$\bar{G}_{D_2} = \{D_2(v_i,0,1), D_2(v_i,1,0) \mid v_i \in \vec{y}_1 \cup \vec{x}_2\}$
$\bar{G}_{D_1} = \{D_1(v_i,0,1), D_1(v_i,1,0) \mid v_i \in \vec{x}_1\}$
where v_i is a new constant (URI) for every $v_i \in F$.
We further define the set of rules $\bar{\mathcal{R}} = \bar{\mathcal{R}}_{000} \cup \bar{\mathcal{R}}_{D_1}$ where
$\bar{\mathcal{R}}_{000} = \big\{\{C(X,Y,Z)\} \Rightarrow \{C(0,0,0)\}; \{D_2(X,Y)\} \Rightarrow \{C(0,0,0)\};$
$\{D_1(X,Y,Z)\} \Rightarrow \{C(0,0,0)\}\big\}$
$\bar{\mathcal{R}}_{D_1} = \big\{\{D_1(V,0,1)\} \Rightarrow \{D_1(V,1,0)\}; \{D_1(V,1,0)\} \Rightarrow \{D_1(V,0,1)\}\big\}$
and finally the set of constraints $\bar{\mathcal{C}} = \bar{\mathcal{C}}_G \cup \bar{\mathcal{C}}_{000} \cup \bar{\mathcal{C}}_F$ with
$\bar{\mathcal{C}}_G = \big\{\{C(0,0,0)\} \Rightarrow \{\bar{t}\} \mid \bar{t} \in \bar{G}\big\}$
$\bar{\mathcal{C}}_{000} = \big\{\{D_1(V,1,0), D_1(V,0,1)\} \Rightarrow \{C(0,0,0)\}\big\}$
$\bar{\mathcal{C}}_F = \big\{\forall\vec{A}_1, \vec{X}, \vec{X}', \vec{Y}, \vec{Y}' \{D_1(v_1, X_1, X_1'), \ldots D_1(v_r, X_r, X_r'), D_2(Y_1, Y_1), \ldots, D_2(Y_s, Y_s')\} \Rightarrow$
$\exists\vec{Z}, \vec{Z}' \{D_2(Z_1, Z_1'), \ldots, D_2(Z_t, Z_t'), C(L^*_{1,1}\, L^*_{1,2}, L^*_{1,3}), \ldots, C(L^*_{n,1}\, L^*_{n,2}, L^*_{n,3})\}\big\}$

where $L^*_{i,j} = \begin{cases} X_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ \bar{X}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ Y_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ \bar{Y}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ Z_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \\ \bar{Z}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \end{cases}$

The idea behind this reduction is as follows: Note that every valid subgraph G' (i.e., every $G' \subset G$ that satisfies $\mathcal{C}$ and for which $G \subseteq \mathcal{R}(G')$ holds) has the following two properties:
(1) G' must not contain $\bar{G}_{000}$ and (2) G' must contain exactly one of $D_1(v_i,0,1)$ and $D_1(v_i,1,0)$ for every $v_i \in \vec{x}_1$. Based on these properties, one can then define a truth assignment on $\vec{x}_1$ from G' by the tuples $D_1(v_i,\cdot,\cdot)$ remaining in G'. Finally, it can be shown that the last tgd in $\mathcal{C}$ is satisfied over some valid subgraph G' iff for the truth assignment on $\vec{x}_1$ mentioned above, $\forall\vec{y}_1\exists\vec{x}_2 \bigwedge_{i=1}^{n} C_i$ holds. This is due to a one-to-one correspondance between homomorhphisms from this tgd to G' and truth assignments on $\bigwedge_{i=1}^{n} C_i$. At this point, we don't prove this formally, but first show the corresponding reduction to instances of RDF graphs, and then give the formal proof.

The result of the above reduction can be translated into an RDF graph. Before we state the formal definition of this problem reduction, we give some ideas how to derive this reduction from the ideas sketched

above. Every tuple $C(X, Y, Z)$ in the instance $\bar{G}$ is replaced by three triples $X\ h_1\ a_{XYZ}\ .\ Y\ h_2\ a_{XYZ}\ .\ Z\ h_3\ a_{XYZ}$ where a_{XYZ} is a unique URI for every combination of $XYZ \in \{0,1\}^3$. The translation of D_1 and D_2 is done similarily. Finally we add triples to G that allow us to access the concrete truth value of any literal, given a truth assignment. The resulting RDF graph $G = G_{01} \cup G_{000} \cup G_{q1} \cup G_{q4} \cup G_{q2}$ is defined as follows: $G_{01} = \{$
$0\ h_1\ a_{001}\ .\ 0\ h_2\ a_{001}\ .\ 1\ h_3\ a_{001}\ .\ 0\ h_1\ a_{010}\ .\ 1\ h_2\ a_{010}\ .\ 0\ h_3\ a_{010}\ .$
$1\ h_1\ a_{100}\ .\ 0\ h_2\ a_{100}\ .\ 0\ h_3\ a_{100}\ .\ 0\ h_1\ a_{011}\ .\ 1\ h_2\ a_{011}\ .\ 1\ h_3\ a_{011}\ .$
$1\ h_1\ a_{101}\ .\ 0\ h_2\ a_{101}\ .\ 1\ h_3\ a_{101}\ .\ 1\ h_1\ a_{110}\ .\ 1\ h_2\ a_{110}\ .\ 0\ h_3\ a_{110}\ .$
$1\ h_1\ a_{111}\ .\ 1\ h_2\ a_{111}\ .\ 1\ h_3\ a_{111}\ \}$
$G_{000} = \{0\ h_1\ a_{000}\,.\,0\ h_2\ a_{000}\,.\,0\ h_3\ a_{000}\}$
$G_{q1} = \{v_i\ q_1\ a_{01}\,.\,v_i\ q_1\ a_{10} \mid i \leq r\}$
$G_{q4} = \{v_i\ q_4\ a_{01}\,.\,v_i\ q_4\ a_{10} \mid r < i \leq r+t\}$
$G_{q2} = \{0\ q_2\ a_{01}\,.\,1\ q_2\ a_{10}\,.\,1\ q_3\ a_{01}\,.\,0\ q_3\ a_{10}\}$
where v_i is a new URI for each $v_i \in V$. With this we derive at the following rules:
$\mathcal{R} = \{\{R\ q_1\ a_{01}\} \Rightarrow \{R\ q_1\ a_{10}\};$
$\{R\ q_1\ a_{10}\} \Rightarrow \{R\ q_1\ a_{01}\};$
$\{S\ P\ O\} \Rightarrow G_{000}\}$
and the following constraints
$\mathcal{C} = \{\{0\ P\ a_{000}\} \Rightarrow \{\tau\} \mid \tau \in G\} \cup$
$\{\{R\ q_1\ a_{10}\,.\,R\ q_1\ a_{01}\} \Rightarrow G_{000}\} \cup$
$\{\forall \vec{A}_1, \vec{X}, \vec{\bar{X}}, \vec{Y}, \vec{\bar{Y}}\, \phi \Rightarrow \exists \vec{A}_2, \vec{Z}, \vec{\bar{Z}}, \vec{R}\, \psi\}$, where
$\vec{A}_1 = A_1, \ldots, A_{r+s}$, $\vec{A}_2 = A_{r+s+1}, \ldots, A_{r+s+t}$,
$\vec{X} = X_1, \ldots, X_r$, $\vec{\bar{X}} = \bar{X}_1, \ldots, \bar{X}_r$,
$\vec{Y} = Y_1, \ldots, Y_s$, $\vec{\bar{Y}} = \bar{Y}_1, \ldots, \bar{Y}_s$,
$\vec{Z} = Z_1, \ldots, Z_t$, $\vec{\bar{Z}} = \bar{Z}_1, \ldots, \bar{Z}_t$,
$\vec{R} = R_1, \ldots, R_n$,
$\phi = \{v_i\ q_1\ A_i\,.\,X_i\ q_2\ A_i\,.\,\bar{X}_i\ q_3\ A_i \mid i \in [r]\} \cup$
$\{v_i\ q_4\ A_i\,.\,Y_j\ q_2\ A_i\,.\,\bar{Y}_j\ q_3\ A_i \mid r+1 \leq i \leq r+s \text{ and } j = i-r\}$ and
$\psi = \{v_i\ q_4\ A_i\,.\,Z_j\ q_2\ A_i\,.\,\bar{Z}_j\ q_3\ A_i \mid r+s+1 \leq i \leq r+s+t \text{ and } j = i-r-s\} \cup$
$\{L^*_{i,1}\ h_1\ R_i\,.\,L^*_{i,2}\ h_2\ R_i\,.\,L^*_{i,3}\ h_3\ R_i \mid i \in [n]\}$

$$\text{with } L^*_{i,j} = \begin{cases} X_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ \bar{X}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ Y_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ \bar{Y}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ Z_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \\ \bar{Z}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \end{cases}$$

This reduction is obviously feasible in LOGSPACE. It therefore remains to show formally that the reduction is correct, i.e. that F is satisfied iff there exists some $G' \subset G$, $G' \neq \emptyset$ s.t. (1) G' satisfies $\mathcal{C}$ and (2) $G \subseteq Cl_{\mathcal{R}}(G')$. (Note that G satisfies $\mathcal{C}$ indeed holds.)

"if"-direction) Assume that there exists a G' that satisfies (1) and (2). Then we define a truth assignment T_1 on $\vec{x}_1$ as follows: $T_1(v_i) = \mathit{true}$ $(i \leq r)$ if the triple $\{v_i\ q_1\ a_{10}\}$ is in G', and $T_1(v_i) = \mathit{false}$ if the triple $\{v_i\ q_1\ a_{01}\}$ is in G'. To see that this gives indeed a well-defined truth assignment, note that neither both triples can be contained in any proper subset of G that satisfies $\mathcal{C}$ (because otherwise, by the second tgd,

G_{000} has to be contained in G', and in turn because of the first tgd, every triple from G is required to be in G', hence $G = G'$), nor can none of them be in G', as one of the two triples is needed to derive the other one when applying $\mathcal{R}$. Hence one of them has to be in G', as otherwise (2) would not be satisfied by G'. For an arbitrary truth assignment T_2 on $\vec{y}_1$, we now define a mapping μ on $\vec{A}_1$, $\vec{X}$, $\vec{\bar{X}}$, $\vec{Y}$, $\vec{\bar{Y}}$:
For $i \leq r$ (i.e. $v_i \in \vec{x}_1$),
$\mu(A_i) = a_{10}$, $\mu(X_i) = 1$, and $\mu(\bar{X}_i) = 0$ if $T_1(v_i) = \mathit{true}$ and
$\mu(A_i) = a_{01}$, $\mu(X_i) = 0$, and $\mu(\bar{X}_i) = 1$ if $T_1(v_i) = \mathit{false}$.
Moreover, for $r < i \leq r + s$ (i.e. $v_i \in \vec{y}_1$),
$\mu(A_i) = a_{10}$, $\mu(Y_{i-r}) = 1$, and $\mu(\bar{Y}_{i-r}) = 0$ if $T_2(v_i) = \mathit{true}$ and
$\mu(A_i) = a_{01}$, $\mu(Y_{i-r}) = 0$, and $\mu(\bar{Y}_{i-r}) = 1$ if $T_2(v_i) = \mathit{false}$.

It can be easily verified that μ is a homomorphism from ϕ to G': Just note that for every $v_i \in \vec{y}_1$ both, the triples $\{v_i\ q_4\ a_{10}\,.\,1\ q_2\ a_{10}\,.\,0\ q_3\ a_{10}\}$ and $\{v_i\ q_4\ a_{01}\,.\,0\ q_2\ a_{01}\,.\,1\ q_3\ a_{01}\}$ are in G', hence regardless of whether $T_2(v_i)$ is true or false, the result of applying μ to ϕ is in G'. Therefore, as G' satisfies $\mathcal{C}$, there exists an extension μ' of μ that maps ψ into G'. It is easy to check that every such extension maps every blank node from $\vec{Z}$ and $\vec{\bar{Z}}$ to a value in $\{0, 1\}$, every blank node in $\vec{A}_2$ to a value in $\{a_{01}, a_{10}\}$ and every blank node in $\vec{R}$ to a value in $\{a_{001}, \ldots, a_{111}\}$ (no $R_i \in \vec{R}$ can be mapped to a_{000}, as, due to the fact that G' satisfies $\mathcal{C}$ and $G' \subset G$, G' cannot contain any of the three triples in G with a_{000} on object position). Moreover, for every $i \in [t]$, $\mu'(Z_i) = 1 - \mu'(\bar{Z}_i)$ holds.

From this, we define a truth assignment T_3 on $\vec{x}_2$: For $i > r + s$ (i.e. $v_i \in \vec{x}_2$), let $T_3(v_i) = \mathit{true}$ if $\mu'(Z_{i-r-s}) = 1$ (hence $\mu'(\bar{Z}_{i-r-s}) = 0$), and $T_3(v_i) = \mathit{false}$ if $\mu'(Z_{i-r-s}) = 0$ (hence $\mu'(\bar{Z}_{i-r-s}) = 1$). Obviously, this truth assignment is well-defined, as μ' is a valid homomorphism, hence it maps each blank node in ψ to exactly one value of G'.

We claim that $T = T_1 \cup T_2 \cup T_3$ is a truth assignment under which $\bigwedge_{i=1}^{n} C_i)$ evaluates to true. To see this, note that for every $i \in [n]$, μ' cannot map all three, $L^*_{i,1}$, $L^*_{i,2}$ and $L^*_{i,3}$ to 0, because, just as stated above, there exists no value in G' for $\mu'(R_i) = a_{xyz}$, such that G' contains the triples $\{0\ h_1\ a_{xyz}\ .\ 0\ h_2\ a_{xyz}\ .\ 0\ h_3\ a_{xyz}\}$ (the only value in G for which this holds, a_{000}, does not appear in G'). Therefore (at least) one of them, say $L^*_{i,j}$ is mapped to 1. Now, by the definition of $L^*_{i,j}$ from $l_{i,j}$, and the definition of T, this translates to the fact that $T(l_{i,j}) = \mathit{true}$ for the following reason: $L^*_{i,j}$ is X_β (resp. Y_β, Z_β) iff $l_{i,j}$ is a positive literal. Hence $T(l_{i,j})$ is the same as the truth value of the variable of $l_{i,j}$, which is true by the above definition. On the other hand, $L^*_{i,j}$ is $\bar{X}_\beta$ (resp. $\bar{Y}_\beta$, $\bar{Z}_\beta$) iff $l_{i,j}$ is a negative literal. But from $\mu'(L^*_{i,j}) = 1$ it follows that $T(v_\alpha) = \mathit{false}$, hence $T(l_{i,j}) = \mathit{true}$. Therefore every clause in F evaluates to true, which proves the case.

"only if"-direction) Assume that F is satisfied. Then there exists a truth assignment T_1 on $\vec{x}_1$ s.t. for all truth assignments T_2 on $\vec{y}_1$ there exists a truth assignment T_3 on $\vec{x}_2$ such that $\bar{F} = \bigwedge_{i=1}^{n} C_i$ is satisfied under $T = T_1 \cup T_2 \cup T_3$. We have to show that then there exists a $G' \subset G$ that satisfies (1) and (2).

Towards this goal, for every $v_i \in \vec{x}_1$, denote with τ_i the triple $\{v_i\ q_1\ a_{10}\}$ if $T_1(v_i) = \mathit{false}$, and $\{v_i\ q_1\ a_{01}\}$ if $T_1(v_i) = \mathit{true}$. (As stated above, by slight abuse of notation, we use v_i to denote both, the variable in F and the corresponding URI in G.) Then we define $G' = G \setminus (\{\tau_i \mid \text{for all } v_i \in \vec{x}_1\} \cup G_{000})$. It is easy to see that G' indeed satisfies (2): As T_1 assigns to every $v_i \in \vec{x}_1$ either true or false, for every $i \leq r$, either $\{v_i\ q_1\ a_{10}\}$ or $\{v_i\ q_1\ a_{01}\}$ is in G'. Therefore, the removed triple can be derived from the present one via $\mathcal{R}$. Moreover, G_{000} can be derived by $\mathcal{R}$ as long as $G' \neq \emptyset$, which is obviously the case.

It remains to show that G' satisfies $\mathcal{C}$.
For the tgds from $\{\{0\ P\ a_{000}\} \Rightarrow \{\tau\} \mid \tau \in G\}$, this is easy to see: $G_{000} \not\subseteq G'$, hence the antecedent of these tgds cannot be mapped to G', because G' no longer contains the URI a_{000}. Also the tgd $\{\{R\ q_1\ a_{10}\,.\,R\ q_1\ a_{01}\} \Rightarrow G_{000}\}$ is obviously satisfied over G', as we have shown before that for no v_i

($i \leq r$), both triples $\{v_i\ q_1\ a_{10}\,.\,v_i\ q_1\ a_{01}\}$ are in G'. But as these are the only triples with q_1 on predicate position, also this tgd is satisfied over G'.
For the tgd $\forall \vec{A}_1, \vec{X}, \vec{\bar{X}}, \vec{Y}, \vec{\bar{Y}}\ \phi \Rightarrow \exists \vec{A}_2, \vec{Z}, \vec{\bar{Z}}, \vec{R}\,\psi$, let μ be an arbitrary homomorphism from ϕ to G'. Note that $\mu(X_i) = 1 - \mu(\bar{X}_i)$, and $\mu(Y_i) = 1 - \mu(\bar{Y}_i)$ holds (for $i \leq r$ resp. $r < i \leq r + s$). We claim that for $i \leq r$ the following relationship between T_1 and μ holds:
$T_1(v_i) = \mathit{true}$ iff $\mu(A_i) = a_{10}$ (i.e. $\mu(X_i) = 1$ and $\mu(\bar{X}_i) = 0$) and
$T_1(v_i) = \mathit{false}$ if $\mu(A_i) = a_{01}$ (i.e. $\mu(X_i) = 0$ and $\mu(\bar{X}_i) = 1$).
This is due to the construction of G', where according to T_1, either $v_i\ q_1\ a_{10}$ or $v_i\ q_1\ a_{01}$ has been removed from G, such that every homomorphism must map A_i to the correct value. Now we define a truth assignment T_2 on $\vec{y}_1$ as follows: For every $r < i \leq r + s$, we define $T_2(v_i) = \mathit{true}$ if $\mu(A_i) = a_{10}$ (i.e. $\mu(Y_{i-r}) = 1$ and $\mu(\bar{Y}_{i-r}) = 0$) and $T_2(v_i) = \mathit{false}$ if $\mu(A_i) = a_{01}$ (i.e. $\mu(Y_{i-r}) = 0$ and $\mu(\bar{Y}_{i-r}) = 1$). It is easy to see that T_2 is indeed a well-defined truth assignment: Every homomorphism must map A_i either to a_{10} or to a_{10}. Therefore, by the assumption that F is satisfied, there must exists some T_3 s.t. $T = T_1 \cup T_2 \cup T_3$ satisfies $\bar{F}$. From T_3, we define an extension of μ' of μ onto $\vec{A}_2, \vec{Z}, \vec{\bar{Z}}, \vec{R}$, s.t. $\mu'(\psi) \subseteq G'$. First, for $\vec{A}_2, \vec{Z}, \vec{\bar{Z}}$ and $i \in \{r + s + 1, \ldots, r + s + t\}$, we define
$\mu'(A_i) = a_{10}$, $\mu'(Z_{i-r-s}) = 1$, and $\mu'(\bar{Z}_{i-r-s}) = 0$ if $T_3(v_i) = \mathit{true}$ and
$\mu'(A_i) = a_{01}$, $\mu'(Z_{i-r-s}) = 0$, and $\mu'(\bar{Z}_{i-r-s}) = 1$ if $T_3(v_i) = \mathit{false}$.
It is again easy to see that μ' indeed maps the triples $\{v_i\ q_4\ A_i\,.\,Z_j\ q_2\ A_i\,.\,\bar{Z}_j\ q_3\ A_i \mid r + s + 1 \leq i \leq r + s + t \text{ and } j = i - r - s\}$ to G'. We further extend μ' to $\vec{R}$ by defining $\mu'(R_i) = a_{xyz}$, where $x = \mu'(L^*_{i,1})$, $y = \mu'(L^*_{i,2})$, and $z = \mu'(L^*_{i,3})$. Finally, we claim that this maps the remaining triples in ψ to G'. As T satisfies $\bigwedge_{i=1}^{n} C_i$, for every clause C_i, $(T(l_{i,1}), T(l_{i,2}), T(l_{i,3})) \neq (\mathit{false}, \mathit{false}, \mathit{false})$, i.e. at least one literal $l_{i,j}$ in every C_i evaluates to true. By the definition of μ' and $L^*_{i,j}$, this means that for every $i \in [n]$, $(\mu'(L^*_{i,1}), \mu'(L^*_{i,2}), \mu'(L^*_{i,3})) \neq (0, 0, 0)$. But for every other combination (x, y, z) with $x, y, z \in \{0, 1\}$, there exists an a_{xyz} in the domain of G', such that every triple in ψ is mapped to a triple in G': That is, there exist triples $\{\mu'(L^*_{i,1})\ h_1\ a_{\mu'(L^*_{i,1})\mu'(L^*_{i,2})\mu'(L^*_{i,3})}\ .\ \mu'(L^*_{i,2})\ h_2\ a_{\mu'(L^*_{i,1})\mu'(L^*_{i,2})\mu'(L^*_{i,3})}\ .\ \mu'(L^*_{i,3})\ h_3\ a_{\mu'(L^*_{i,1})\mu'(L^*_{i,2})\mu'(L^*_{i,3})}\}$, which proves the case.

To justify our claim that this reduction works for MINI-RDF$^{\models}$ as well, please note that G contains no blank nodes. As all rules in $\mathcal{R}$ are save, also $Cl_{\mathcal{R}}(G)$ cannot contain any blank node. □

Lemma 3.5. *The problems* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$*, where both, $\mathcal{R}$ and $\mathcal{C}$ are considered as fixed, are* NP*-hard.*

Proof. First consider the case of MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$. We show that even for $\mathcal{C} = \mathcal{R} = \emptyset$, the problem is NP-hard: In this case, $(G, \mathcal{R}, \mathcal{C})$ is a positive instance of MINI-RDF$^{\models}$ iff there exists some $G' \subset G$ s.t. $G' \models G$. This is exactly the case if G is not lean. As deciding whether a graph is lean is a coNP-complete problem [15], the NP-hardness follows immediately.

Now consider MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$, and recall the reduction given in the main body of the text:
We consider a fixed set of rules and tgds as
$\mathcal{R} = \{\{X'\ \mathit{in}\ I\,.\,X\ \mathit{active}\ I\} \Rightarrow \{X'\ \mathit{active}\ I\}\}$ and
$\mathcal{C} = \{\{X\ \mathit{active}\ I\,.\,X\ \mathit{in}\ J\} \Rightarrow \{X\ \mathit{active}\ J\};$
$\{X\ \mathit{clash}\ X'\,.\,X\ \mathit{active}\ I\,.\,X'\ \mathit{active}\ I'\,.\,Y\ \mathit{in}\ J\} \Rightarrow \{Y\ \mathit{active}\ J\}\}$

Now let an arbitrary instance of 3-SAT be given by the formula $F = \bigwedge_{i=1}^{n} C_i$, where $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ and $l_{i,j}$ are literals. W.l.o.g., we assume that every variable appears negated and unnegated in F, and denote the number of variables occurring in F with m. By slight abuse of notation, we use x_i to denote

both, variables in F and URIs in G. Then we construct an RDF graph G as
$G = \{l^*_{i,j}\ in\ c_i \mid i \in [n], j \in [3]\} \cup$
$\{l^*_{i,j}\ active\ c_i \mid i \in [n], j \in [3]\} \cup$
$\{x_\alpha\ clash\ \bar{x}_\alpha \mid \alpha \in [m]\}$.
Thereby we introduce new URIs c_i (for every clause C_i) and $x_\alpha, \bar{x}_\alpha$ (for every variable x_α in F), and define $l^*_{i,j} = x_\alpha$ (resp. $\bar{x}_\alpha$) if $l_{i,j} = x_\alpha$ (resp. $\neg x_\alpha$).

The reduction is obviously feasible in LOGSPACE. Before proving its correctness, we first sketch its intuition: First we observe that none of the triples containing in or $clash$ on predicate position can be removed from G, as they cannot be derived using $\mathcal{R}$. Now the basic idea is that the triples of the form $l^*_{i,j}\ active\ c_i$, that remain in the subgraph $G' \subset G$ encode a truth assignment T on the variables in F. More concretely, they encode which literals in each clause evaluate to $true$ under T (namely exactly those literals whose corresponding triples remain in G'). To express this, and to make sure that this indeed gives a valid truth assignment, we use the triples of the form $l^*_{i,j}\ in\ c_i$ to encode which variable occurs in which clause, and whether it occurs negated or unnegated (by using URIs x_α and $\bar{x}_\alpha$, resp.). The triples $x\ clash\ \bar{x}$ encode that while x represents an unnegated occurence of variable x, $\bar{x}$ encodes a negated occurence. Hence, in no valid proper subgraph G', x and $\bar{x}$ are both allowed to be active (as expressed by the second tgd, which requires in this case that $G' = G$), as this would mean that $T(x) = T(\neg x)$. The only rule in $\mathcal{R}$ ensures that in every clause, at least one literal evaluates to $true$, since otherwise none of the triples $x_\alpha\ active\ c_i$ for this clause (which are present in G) can be derived from G'.

It remains to show that the reduction is indeed correct, i.e. that F is satisfiable iff there exists $\emptyset \neq G' \subset G$ s.t. (1) $G \subseteq Cl_\mathcal{R}(G')$ and (2) G' satisfies $\mathcal{C}$.

"if"-direction) Assume such a G' exists. As observed above, G' can differ from G only by triples containing the URI $active$ in predicate position, as these are the only triples derivable by $\mathcal{R}$. We define a truth assignment T on the variables of F as follows: For every variable x_j in F, $T(x_\alpha) = true$ if $\{x_\alpha\ active\ c_i\} \subseteq G'$ for some c_i, $T(x_\alpha) = false$ if $\{\bar{x}_\alpha\ active\ c_i\} \subseteq G'$ for some c_i, and $T(x_\alpha) = true$ if neither of them is in G'.

It remains to show that this truth assignment is well-defined (i.e. every variable gets assigned exactly one truth value) and that it indeed satisfies F. By the definition of T, obviously every variable gets some truth value assigned. To see that this truth value is unique, assume to the contrary that this is not the case. Then, for some variable x_α, both, $\{x_\alpha\ active\ c_i\}$ and $\{\bar{x}_\alpha\ active\ c_k\}$ must be in G'. By construction of G, $\{\bar{x}_\alpha\ clash\ x_\alpha\}$ is in G (hence G'). Therefore the antecendent of the second tgd is satisfied for every variable occurence in every clause. This requires all triples $\{\_\ active\ \_\}$ from G to be in G' as well, as otherwise G' would not satisfy $\mathcal{C}$. But as those triples are the only ones that can be derived by $\mathcal{R}$, this implies that $G' = G$, which is a contradiction to the assumption $G' \subset G$. Therefore T is a well-defined truth assignment.

It finally remains to show that T satisfies F. I.e., let C_i be an arbitrary clause in F. We have to show that C_i is $true$ under T. Towards this goal, we first prove that there exists at least one triple $\{X\ active\ c_i\}$ in G' (where X is either some x_α or $\bar{x}_\alpha$). To see that this is indeed the case, assume to the contrary that no such triple remains in G'. It is easy to see, that then also $Cl_\mathcal{R}(G')$ cannot contain any triple with $active$ in predicate position and c_i in object position, as the antecedent of the rule is never satisfied for c_i. But then G' does not satisfy (1), which gives a contradiction to our assumption. Using this, it is now easy to show that the literal encoded by X evaluates to $true$ under T. If the subject position of the triple contains some URI x_α, then the corresponding variable occurs unnegated in C_i and gets the truth value $true$ assigned by T, and if the subject position of the triple contains $\bar{x}_\alpha$, then the corresponding variable occurs negated in C_i, getting the truth value $false$ assigned by T. In both cases, the literal evaluates to $true$, which proves the case.

"only-if"-direction) Assume now that there exists a truth assignment that satisfies F. We have to show that then there exists a subgraph $G' \subset G$ satisfying (1) and (2). Denoting the truth assignment with T, we first define $G' \subset G$: G' contains all triples from G, except of the following. For every clause C_i and every literal $l_{i,j}$ that appears in C_i, if $l_{i,j}$ is of the form $\neg x_\alpha$ and $T(x_\alpha) = \mathit{true}$, then remove the triple $\bar{x}_\alpha \ \mathit{active}\ c_i$. If on the other hand, $l_{i,j}$ is of the form x_α and $T(x_\alpha) = \mathit{false}$, then the triple $x_\alpha \ \mathit{active}\ c_i$ is not in G'. Otherwise the triple representing $l_{i,j}$ remains in G'. From the assumptions (every variable occurs at least once negated and unnegated in F), it follows immediately that G' is indeed a proper subgraph of G. It remains to show that G' satisfies (1) and (2).

(1): Because T satisfies F, in every clause C_i at least one literal evaluates to true, say $l_{i,j}$. Hence, if $l_{i,j}$ is of the form x_α, the triple $\{x_\alpha \ \mathit{active}\ c_i\}$ remains in G'. If on the other hand $l_{i,j}$ is of the form $\neg x_\alpha$, the triple $\{\bar{x}_\alpha \ \mathit{active}\ c_i\}$ remains in G'. Therefore every triple $\{X \ \mathit{active}\ c_i\}$ can be derived from G' by $\mathcal{R}$.

(2): To see that the first tgd is satisfied, note that by our definition of G', for every pair x_α and $\bar{x}_\alpha$, either all triples $\{x_\alpha \ \mathit{active}\ c_i\}$ (for every c_i where x_α occurs in positively), resp. $\{\bar{x}_\alpha \ \mathit{active}\ c_i\}$ (for all c_i where x_α occurs negatively) are removed from G, or none. Hence if G' contains the triple $\{x_\alpha \ \mathit{active}\ c_i\}$, then it also contains all other triples $\{x_\alpha \ \mathit{active}\ c_r\}$ from G, and therefore the tgd is satisfied.

For the second tgd, note that T is a well-defined truth assignment on all variables in F. Hence for every variable x_α, either all triples $\{x_\alpha \ \mathit{active}\ \_\}$ or all triples $\{\bar{x}_\alpha \ \mathit{active}\ \_\}$ are removed from G for G'. But because the only triples containing the clash predicate are of the form $\{x_\alpha \ \mathit{clash}\ \bar{x}_\alpha\}$, the antecedent of the second tgd cannot be mapped to G', hence it is trivially always satisfied, which proves the case. □

Lemma A.1. *Both,* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$*, for arbitrary sets $\mathcal{R}$ and $\mathcal{C}$ can be solved in Σ_3^P.*

Proof. The following nondeterministic polynomial algorithm with a Π_2^P-oracle algorithm decides MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$:
We first point out, that every rule with more than one triple in the consequent can be replaced in some preprocessing step by several rules with the same antecendent, each of them containing in its consequent exactly one triple from the original consequent.

(1) Guess a subgraph $G' \subset G$
(2) Check (by a Π_2^P-oracle) if G' satisfies G (otherwise return "No")
(3) Set $\hat{G}_0 = G'$ and $k = |AD|^3$,
  where $AD = U_G \cup U_\mathcal{R} \cup B_G \cup B_\mathcal{R} \cup L_G \cup L_\mathcal{R}$ denotes the active domain.
  Further let r_0 encode the "do nothing" rule $r_0 \colon \emptyset \Rightarrow \emptyset$.
(4)For $j = 1, \ldots, k$:
  a) Guess a rule $r_i \colon G_i \Rightarrow \{t_i\}$ from $\mathcal{R} \cup \{r_0\}$
  b) Guess a mapping $\mu \colon G_i \to \hat{G}_{j-1}$
    (from the rule body G_i to the intermediate graph)
  c) Check whether μ is a homomorphism $G_i \to \hat{G}_{j-1}$. If not return "No"
  d) $\hat{G}_j = \hat{G}_{j-1} \cup \{\mu(t_i)\}$
(4) Guess a mapping $G \to \hat{G}_k$
(5) If it is a valid homomorphism, return "Yes", otherwise return "No"

Step (2) can be solved by a Π_2^P-oracle: Just guess a mapping from the antecedent to G. If it is a homomorphism, then use the NP-oracle to check whether there exists an extension of this mapping that maps the consequent to G too.

For MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$, in steps (4) and (5), instead of testing $Cl_{\mathcal{R}}(G') \models G$, one needs to test whether $G \subseteq Cl_{\mathcal{R}}(G')$, which is obviously not harder. □

Lemma A.2. *Both,* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$*, for arbitrary sets of rules* $\mathcal{R}$ *and sets* $\mathcal{C}$ *of b-bounded tgds, are in* NP.

Proof. The problems can be decided by the algorithms depicted in the proof of Lemma A.1, but (by [11]/Proposition 3) step (2) now requires only polynomial time, hence no call to an oracle is needed. □

Lemma A.3. *The problem* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ *can be decided in* PTIME *if all rules in* $\mathcal{R}$ *are b-bounded and there are no constraints (i.e.* $\mathcal{C} = \emptyset$*).*

Proof. The problem can be decided by testing for every triple $t \in G$, whether it can be removed from G: For every $t \in G$, test whether $t \in Cl_{\mathcal{R}}(G \setminus \{t\})$. If this is the case for some triple, then there obviosuly exists some smaller graph, hence the answer is *Yes*. Otherwise, as no triple in G can be derived via the rules in $\mathcal{R}$ from the remaining triples, the answer is *No*. By [11]/Proposition 9, the check requires only polynomial time. □

Lemma A.4. *The problems* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$*, for arbitrary sets* $\mathcal{R}$ *and no constraints (i.e.* $\mathcal{C} = \emptyset$*) are* NP*-hard.*

Proof. The hardness follows immediately from the fact that testing whether some datalog rule $r_i\colon G_i \Rightarrow \{t_i\}$ is applicable (i.e. whether there exists a homomorphism $G_i \rightarrow G$) is already NP-complete [15]. □

Lemma A.5. *The problem* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *for a fixed set* $\mathcal{R}$ *and no constraints (i.e.* $\mathcal{C} = \emptyset$*) is* NP*-hard.*

Proof. Follows immediately from the proof of Lemma 3.5 for MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$. □

B Full proofs for Section 4

Theorem 4.1. *For a set* $\mathcal{R}$ *of b-bounded rules (for fixed b), the problem* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *is* NP*-complete while* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ *is in* PTIME*.*

Proof. For the PTIME-*membership* of RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$, we consider the following algorithm to test whether $\mathcal{R}$ contains redundant rules:

(1) Compute $G_{clos} = Cl_{\mathcal{R}}(G)$
(2) For every $r \in \mathcal{R}$
  (2a) set $\mathcal{R}' = \mathcal{R} \setminus \{r\}$
  (2b) Compute $G' = Cl_{\mathcal{R}'}(G)$
  (2c) If $G' = G_{clos}$ then return "yes"
(3) return "no".

As $\mathcal{R}$ is b-bounded, the closure of G under (a subset of) $\mathcal{R}$ can be computed in polynomial time. Hence, the entire algorithm works in polynomial time. For the RDF-RULEMIN$^{\models}(G, \mathcal{R})$ problem, step (2c) is replaced

by the check if $G' \models G_{clos}$ holds. Moreover, the loop in step (2) over all $r \in \mathcal{R}$ is now replaced by guessing an appropriate rule $r \in \mathcal{R}$ (to avoid multiple entailment checks).

The NP-hardness of the RDF-RULEMIN$^{\models}(G, \mathcal{R})$ problem is shown by reduction from 3-Colorability. Let an arbitrary instance of 3-Colorability be given by the graph (V, E) and let $V = \{v_1, \ldots, v_n\}$. We invent URIs d, e and blank nodes $X = \{x_1, \ldots, x_n\}$. Then the RDF graph G and rule set $\mathcal{R}$ are defined as follows:

$G = \{0\ d\ 1\,.\,0\ d\ 2\,.\,1\ d\ 2\,.\,1\ d\ 0\,.\,2\ d\ 0\,.\,2\ d\ 10\ e\ 1\,.\,0\ e\ 2\,.\,1\ e\ 2\,.\,1\ e\ 0\,.\,2\ e\ 0\,.\,2\ e\ 1\} \cup$
$\{x_\alpha\ d\ x_\beta \mid (v_\alpha, v_\beta) \in E\}$.

$\mathcal{R} = \{r\}$ with $r = \{Z_1\ d\ Z_2\} \Rightarrow \{Z_1\ e\ Z_2\}$.

Clearly, this reduction is feasible in polynomial time. For its correctness, we observe that $Cl_{\mathcal{R}}(G) = G \cup \{x_\alpha\ e\ x_\beta \mid (v_\alpha, v_\beta) \in E\}$. It remains to show that $G \models Cl_{\mathcal{R}}(G)$ holds (i.e., r is redundant) iff (V, E) is 3-colorable:

First suppose that $G \models Cl_{\mathcal{R}}(G)$ holds, i.e., there exists a homomorphism $h\colon Cl_{\mathcal{R}}(G) \to G$. Clearly, this homomorphism must map every triple $x_\alpha\ e\ x_\beta$ to one of the triples $0\ e\ 1\,.\,0\ e\ 2\,.\,1\ e\ 2\,.\,1\ e\ 0\,.\,2\ e\ 0\,.\,2\ e\ 1$. But then we immediately get a valid 3-coloring f of (V, E) by defining $f(v_i) = h(x_i)$ for every $i \in \{1, \ldots, n\}$.

Conversely, suppose that there exists a valid 3-coloring f of (V, E). Then we can define a mapping h on the blank nodes in X by setting $h(x_i) = f(v_i)$. It is easy to check that h is indeed a homomorphism from $Cl_{\mathcal{R}}(G)$ to G. □

Theorem 4.2. *For arbitrary rules,* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ *is* coNP*-hard and in* Δ_2^P *while* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *is* NP*-hard,* coNP*-hard, and in* Δ_2^P.

Proof. We first prove the Δ_2^P-membership. The RDF-RULEMIN$^{\models}(G, \mathcal{R})$ problem can be decided by the following polynomial-time algorithm with NP-oracle:

(1) For every rule $r \in \mathcal{R}$ (set $\mathcal{R}' = \mathcal{R} \setminus \{r\}$):
  (2) set $G'_{clos} = G_{clos} = G$
  (3) For every $t \in AD^3$:
    (a) For every $r \in \mathcal{R}'$:
      (i) test whether $t \in r(G'_{clos})$ (by a call to an NP-oracle)
      (ii) if yes, set $G'_{clos} = G'_{clos} \cup \{t\}$
    (b) For every $r \in \mathcal{R}$:
      (i) test whether $t \in r(G_{clos})$ (by a call to an NP-oracle)
      (ii) if yes, set $G_{clos} = G_{clos} \cup \{t\}$
  (4) if G'_{clos} or G_{clos} was changed in (3) goto (3)
  (5) if $G'_{clos} \models G_{clos}$ (test using NP-oracle), return "Yes".
(6) return "No".

Where $AD = U_G \cup U_{\mathcal{R}} \cup B_G \cup B_{\mathcal{R}} \cup L_G \cup L_{\mathcal{R}}$ denotes the active domain.

For the RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ problem, in step (5), the condition has to be replaced by $G'_{clos} = G_{clos}$, which obviously fits into polynomial time.

The NP-hardness of the RDF-RULEMIN$^{\models}(G, \mathcal{R})$ problem carries over from Theorem 4.1. It remains to prove the coNP-hardness of RDF-RULEMIN$^{\models}(G, \mathcal{R})$ and RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$. The proof is by reduction from co-3-Colorability. Let an arbitrary instance of co-3-Colorability be given by the graph (V, E) and let $V = \{v_1, \ldots, v_n\}$. Then we construct the RDF graph G and rule set $\mathcal{R}$ as follows:

$G = \{0\ e\ 1\ .\ 0\ e\ 2\ .\ 1\ e\ 2\ .\ 1\ e\ 0\ .\ 2\ e\ 0\ .\ 2\ e\ 1\}$
$\mathcal{R} = \{r\}$ with $r = \{X_\alpha\ e\ X_\beta \mid (v_\alpha, v_\beta) \in E\} \Rightarrow \{X_1\ e\ X_1\}$.

Clearly, this reduction is feasible in polynomial time. For the correctness, we observe that $Cl_\mathcal{R}(G) \subseteq G \cup \{0\ e\ 0\ .\ 1\ e\ 1\ .\ 2\ e\ 2\}$. In other words, $Cl_\mathcal{R}(G)$ contains no blank nodes. Hence, entailment coincides with the superset-relation. It therefore suffices to show that the graph (V, E) is *not* 3-colorable iff $Cl_\mathcal{R}(G) = G$ (i.e., r is redundant). Equivalently, we have to show that (V, E) is 3-colorable iff $Cl_\mathcal{R}(G) \supset G$.

Clearly, $Cl_\mathcal{R}(G) \supset G$ holds, if rule r fires, i.e., there exists a homomorphism h from the body of r to G. From h we immediately get the 3-coloring f of (V, E) by defining $f(v_i) = h(x_i)$ for every $i \in \{1, \ldots, n\}$. Conversely, if there exists a valid 3-coloring f of (V, E), we immediately get the homomorphism from the body of r to G by setting $h(X_i) = f(v_i)$ for every $i \in \{1, \ldots, n\}$. □

C Full proofs for Section 5

As mentioned in the main body of the text, we will show that the upper bounds still hold, even if the conjunctive queries are allowed to contain negation in the body. Therefore, we first introduce the notions of (U)CQs$^\neg$.

For extending (U)CQs to *(U)CQs*$^\neg$ (i.e. allowing for negation), given a query q, we partition $body(q)$ into two sets, $pos(q)$ and $neg(q)$. Intuitively, $pos(q)$ encodes positive information that must be satisfied by G, while G must not contain the negative information encoded by $neg(q)$. In the following, we assume all (U)CQs$^\neg$ to be safe, that is, all blank nodes appearing in $head(q)$ and $neg(q)$ must also appear in $pos(q)$. The result of a CQ$^\neg$ q over some RDF graph G is defined as the set $q(G) = \{(\vec{x}) \mid \text{for all } x_i \in \vec{x}\colon x_i \in U_G B_G L_G U_q L_q$ and there exists a homomorphism $\tau\colon B_q \to U_G B_G L_G$ s.t. $\tau(pos(q)) \subseteq G$ and for all $t \in \tau(neg(q)), t \notin G$ and $\tilde{x} = \tau(\tilde{X})\}$ The result of a UCQ$^\neg$ is the union of the results of its CQs$^\neg$.

In the following, when considering a CQ or UCQ q (i.e. $neg(q) = \emptyset$, we use the expressions $body(q)$ and $pos(q)$ interchangable.

Lemma C.1. *The problem* MINI-RDF$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$*, for b-bounded* $\mathcal{R}$*,* $\mathcal{C} = \emptyset$*, and body-b-bounded* $\mathcal{Q}$ *can be solved in* PTIME*. The problem remains in* PTIME *even if* $\mathcal{Q}$ *is a set of body-b-bounded UCQs*$^\neg$*.*

Proof. The problem can be decided by the following algorithm:

(1) Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
(2) For every $t \in G$:
  (2a) Set $G' = G \setminus \{t\}$
  (2b) Compute $\hat{G'} = Cl_{\mathcal{R}}(G')$
  (2c) For every $q \in \mathcal{Q}$, test if $q(\hat{G}) = q(\hat{G'})$

Then G can be minimised, if for some $t \in G$, step (2c) returns *"Yes"*.

Under the restrictions assumed on the input, this algorithm runs in polynomial time: By [11, Proposition 9], $Cl_{\mathcal{R}}(G)$ (and therefore also $Cl_{\mathcal{R}}(G')$) can be computed in polynomial time. As the size of $Cl_{\mathcal{R}}(G)$ (resp. $Cl_{\mathcal{R}}(G')$) is polynomially bounded in the size of the input, and because the body of each $q \in \mathcal{Q}$ is b-bounded, also step (2c) fits into polynomial time, as at most $|AD|^{3*b}$ homomorphisms need to be checked for every $q \in \mathcal{Q}$ (where AD denotes the active domain).

In the case that $\mathcal{Q}$ is a set of UCQs, just note that every UCQ $Q \in \mathcal{Q}$ contains at most polynomial many CQs q, each of them creating at most a polynomial big result set, such that the algorithm still fits into PTIME, i.e. replace step (2c) by
(2c) For every $Q \in \mathcal{Q}$, test if $Q(\hat{G}) = Q(\hat{G'})$.

To see that step (2c) still fits into PTIME, even if we allow for UCQs$^\neg$, consider the following algorithm:

(A) For every $Q \in \mathcal{Q}$
  (A1) Set $A = A' = \emptyset$
  (A2) For every $q \in Q$
    (A3) For every homomorphism $\tau\colon pos(q) \to \hat{G}$
      (A4) If for every $t \in \tau(neg(q))\colon\ t \notin \hat{G}$, set $A = A \cup \{\tau(head(q))\}$
    (A5) For every homomorphism $\tau\colon pos(q) \to \hat{G'}$
      (A6) If for every $t \in \tau(neg(q))\colon\ t \notin \hat{G'}$, set $A' = A' \cup \{\tau(head(q))\}$
    (A7) For every homomorphism $\tau\colon pos(q) \to \hat{G}$
  (A8) Test if $A = A'$. If not, return *"No"*, otherwise test next Q.
(B) return *"Yes"*

To see that this algorithm indeed runs in polynomial time, note that in steps (A3) and (A5), only polynomial many homomorphisms need to be checked (the number is bound at most by AD^{3b}). Therefore also (A7) obviously fits into polynomial time. □

Lemma C.2. *The problem* MINI-RDF$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$*, where $\mathcal{R}$, $\mathcal{C}$, and $\mathcal{Q}$ are body-b-bounded can be solved in* NP*. The problem remains in* NP*even if $\mathcal{Q}$ is a set of body-b-bounded UCQs$^\neg$.*

Proof. The problem can be decided by the following nondeterministic algorithm:
(1) Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
(2) Guess a subset $G' \subset G$
(3) Check if G' satisfies $\mathcal{C}$ (if not, return *"No"*)
(4) Compute $\hat{G'} = Cl_{\mathcal{R}}(G)$
(5) For every $q \in \mathcal{Q}$, test if $q(\hat{G}) = q(\hat{G'})$
(Return *"Yes"* if it holds, otherwise return *"No"*)
If for some guess G', *"Yes"* is returned, then G can be minimised.

By the same arguments as in the proof of Lemma C.1, the steps (1), (4) and (5) require only polynomial time. Step (3) is in polynomial time because of [11, Proposition 3], and step (2) introduces the nondeterminism.

Again by the same arguments as in the proof of Lemma C.1, the result holds for body-b-bounded UCQs and body-b-bounded UCQs$^\neg$ as well. □

Lemma C.3. *The problem* MINI-RDF$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$*, for arbitrary $\mathcal{R}$, b-bounded $\mathcal{C}$, where all CQs in $\mathcal{Q}$ have bounded head arity can be solved in Δ_2^P. The problem remains in Δ_2^P even if $\mathcal{Q}$ is a set of bounded head arity UCQs$^\neg$.*

Proof. The following deterministic algorithm, using an NP-oracle, decides the problem and requires only polynomial time:
(1) Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
(2) For every $q \in \mathcal{Q}$, compute and store $q(\hat{G})$
(3) In the oracle
  (3a) Guess a $G' \subset G$
  (3b) For every $q(\hat{G})$ and every $t \in q(\hat{G})$
    (I) Guess a derivation sequence for $\mathcal{R}$ on G' $^{(*)}$,
      denote the result of this sequence by $\hat{G'}$.
    (II) For the homomorphism $\tau\colon head(q) \to t$, (s.t. $\tau(head(q)) = t$)
      guess an extension $\tau'\colon body(q) \to \hat{G'}$, and
      check whether $\tau'(body(q)) \subset \hat{G'}$
      (if not, return *"No"*, otherwise try next t, resp. next $q(\hat{G})$)
  (3c) Return *"Yes"*

The program then returns the result of the oracle call. For a justification that step (1) fits into Δ_2^P, we refer to the proof of Theorem 4.2.

To see that step (2) fits into Δ_2^P, consider the following algorithm: For every $t \in |AD|^b$ (where b is the arity of the head of q), guess a homomorphism $\tau\colon body(q) \to \hat{G}$ s.t. $\tau(head(q)) = t$. If such a τ exists, add

t to $q(\hat{G})$. As we assume all $q \in \mathcal{Q}$ to have bounded head arity, b is a constant, hence $|AD|^b$ is polynomial in the input, which allows the above algorithm to be performed in polynomial time with a NP-oracle. Therefore also the size of each $q(\hat{G})$ is at most polynomial in the input.

Concerning step (I) in the oracle, we comment on the $^{(*)}$ mark: By guessing a derivation sequence, we mean (just as in the proof of Lemma A.1) to guess a sequence $r_1, \ldots r_n$ of n rules from $\mathcal{R}$ (where the same rule may appear more than once in the sequence), and for each selection r_i guess a homomorphism from the rule body to the graph derived from G' by application of $r_1, \ldots, r_{i-1}$. (For a more formal description, see step (3) in the proof of Lemma A.1.) Note that although we cannot be sure that the final result of such a derivation sequence is indeed $Cl_{\mathcal{R}}(G')$, we know that every sequence return a subset of $Cl_{\mathcal{R}}(G')$. and that at least one of the guessed sequences indeed gives $Cl_{\mathcal{R}}(G')$. As $Cl_{\mathcal{R}}(G) \subseteq AD^3$, it suffices to check all sequences of length $n = |AD|^3$.

To see that the result holds in the case where $\mathcal{Q}$ is a set of UCQ as well, just note the following:
In step (2) we compute $Q(\hat{G})$ for every UCQ $Q \in \mathcal{Q}$ (by the same algorithm as sketched above for CQs, just that we guess a $q \in Q$ together with the homomorphism).
In step (II), we can either guess $q \in Q$ together with the extension τ', or we can test all $q \in Q$ after each other (i.e. guess one extension τ' for each of them).

Finally, to extend the algorithm also to UCQ$^\neg$, in steps (2) and (II), for every guess of a homomorphism τ (resp. τ'), we do not only check $\tau(pos(q)) \subseteq \hat{G}$ (resp. $\tau'(pos(q)) \subseteq \hat{G}'$), but also whether $\tau(neg(q)) \cap \hat{G} = \emptyset$ (resp. $\tau'(neg(q)) \cap \hat{G}' = \emptyset$). Obviously, this fits into the oracle as well. □

Lemma C.4. *The problem* MINI-RDF$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$*, for arbitrary* $\mathcal{R}$*,* $\mathcal{C} = \emptyset$ *where* $\mathcal{Q}$ *may contain arbitrary CQs can be solved in* Π_2^P*. The problem remains in* Π_2^P *even if* $\mathcal{Q}$ *is a set of arbitrary UCQs*$^\neg$*.*

Proof. We prove the lemma by devising a nondeterministic algorithm that decides the co-problem (that is, whether G is already minimal w.r.t. $\mathcal{R}$, $\mathcal{C}$ and $\mathcal{Q}$), using a coNP-oracle, in polynomial time.

Let $G = \{t_1, \ldots, t_n\}$
For each $i \in \{1, \ldots, n\}$, let $G'_i = G \setminus \{t_i\}$
(1) Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
(2) Compute $\hat{G}_i = Cl_{\mathcal{R}}(G_i)$ for $i \in [n]$
(3) Guess n (not necessarily distinct) queries $q_i \in \mathcal{Q}$
(4) Guess n homomorphisms $\tau_1, \ldots, \tau_n$, with $\tau_i \colon body(q_i) \to \hat{G}$
(5) Check $\forall i \in [n]$ by a call to a coNP-oracle (i.e. using n calls)
that $\tau_i(head(q_i)) \notin q(\hat{G}_i)$, i.e. check that for all homomorphisms
$\tau'_i \colon body(q_i) \to \hat{G}_i$ with $\tau'_i(body(q_i)) \subseteq \hat{G}_i$, whether
$\tau'_i(head(q_i)) \neq \tau(head(q_i))$
(6) G is minimal, iff all n oracle calls return *"yes"*

The intuition of this is as follows. After computing the closure of G and all "candidates" G_i, we guess for every G_i some query a and an answer over $\hat{G}$ to this query, such that the answer cannot be created by q over $\hat{G}_i$. If this is the case, G cannot be further minimised.

To see that this nondeterministic algorithm indeed runs in polynomial time, just note that we already discussed in the previous proof, that the computation of $\hat{G}$ and $\hat{G}_i$ fits into Δ_2^P, and that the size of $\hat{G}$ and each $\hat{G}_i$ is at most polynomial in the size of the input.

Concerning an extension of this algorithm to UCQs, it suffices to replace step (3) by
(3) Guess $Q \in \mathcal{Q}$ and $q \in Q$

and in step (5) to check that $\tau_i(head(q_i)) \notin q(\hat{G}')$, which means that in the coNP oracle one checks for all $q \in Q$ that $q(\hat{G}')$ does not create $\tau_i(head(q_i))$. Again, this can either be done sequentially, for each $q \in Q$ after the other, or in parallel by first guessing $q \in Q$ and then the corresponding homomorphism.

Towards the extension to UCQs$^\neg$, we just note that for every guessed homomorphism τ from the body of some query q into some graph $\bar{G}$, instead of just checking whether $\tau(pos(q)) \subseteq \bar{G}$, we also have to test that $\tau(pos(q)) \cap \bar{G} = \emptyset$. However, this additional check, just as the previous one, obviously fits into polynomial time. □

Lemma C.5. *The problem* MINI-RDF$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$*, for arbitrary sets $\mathcal{R}$ and $\mathcal{C}$, where $\mathcal{Q}$ may contain arbitrary CQs can be solved in Σ_3^P. The problem remains in Σ_3^P even if $\mathcal{Q}$ is a set of arbitrary UCQs$^\neg$.*

Proof. The following nondeterministic algorithm with a Π_2^P-oracle decides the problem:

(1) Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
(2) Guess a $G' \subset G$
(3) Check if G' satisfies $\mathcal{C}$ (otherwise return *"No"*)
(4) Compute $\hat{G}' = Cl_{\mathcal{R}}(G')$
(5) For every $q \in \mathcal{Q}$, check by a call to the oracle, whether
  for every homomorphism $\tau\colon body(q) \to \hat{G}$
  there exists an extension τ' of the homomorphism defined
  by $\tau(head(q))$ to $\tau'\colon body(q) \to \hat{G}'$
  (i.e. whether every result in $q(\hat{G})$ can be also derived over $\hat{G}'$)

To see that this algorithm indeed runs in (nondeterministic) polynomial time, we note that step (1) and step (4) fit into Δ_2^P (see e.g. the proof of Theorem 4.2), step (3) can be decided by a Π_2^P-oracle, and also step (5) can be obviously done by a Π_2^P-oracle.

The algorithm can be extended such that it also works if $\mathcal{Q}$ is a set of UCQs, without changing its complexity. Therefore it suffices to replace step (5) by

(5) For every $Q \in \mathcal{Q}$, check by a call to the oracle, whether for every $q \in Q$
  and every homomorphism $\tau\colon body(q) \to \hat{G}$
  there exists an $q \in Q$ and an extension τ' of the homomorphism defined
  by $\tau(head(q))$ to $\tau'\colon body(q) \to \hat{G}'$
  (i.e. whether every result in $q(\hat{G})$ can be also derived over $\hat{G}'$)

Obviously, this still can be decided by a Π_2^P-oracle. One just needs to guess $q \in Q$ together with the homomorphism.

To further deal with UCQs$^\neg$, step (5) needs to be adopted such that for every guessed homomorphism τ (resp. τ'), it is checked whether $\tau(pos(q)) \subset \hat{G}$ (resp. τ', $\hat{G}'$) and $\tau(neg(q)) \cap \hat{G}$ (resp. τ', $\hat{G}'$) $= \emptyset$. □

Lemma C.6. *The problem* MINI-RDF$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$*, when both, $\mathcal{R}$ and $\mathcal{C}$ are fixed and $\mathcal{Q}$ contains only body-b-bounded CQs is* NP*-hard, already if $\mathcal{Q}$ contains a single CQ only.*

Proof. The hardness follows directly from the observation that the hardness results from MINI-RDF$^\subseteq$ carry over to this setting, as already expressed in the main body of the text by the observation that MINI-RDF$^\subseteq$ corresponds to the special case of MINI-RDF$^{\subseteq, CQ}$ where $\mathcal{Q} = \{\{S\ P\ O\} \to ans(S, P, O)\}$. □

Lemma C.7. *The problem* MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$*, when* $\mathcal{R}$ *is fixed,* $\mathcal{C}=\emptyset$*, and the CQs in* $\mathcal{Q}$ *may have arbitrary BGPs in their bodies but have bounded head arity, is* NP*-hard, already if* $\mathcal{Q}$ *contains a single CQ only.*

Proof. The NP-hardness of the problem can be even shown for $\mathcal{C}=\mathcal{R}=\emptyset$. Given a graph G, it is not lean exactly if the graph G can be reduced w.r.t. the CQ $q\colon G \rightarrow ans()$ (and $\mathcal{C}=\mathcal{R}=\emptyset$). This obviously holds as q trivially returns $true$ over G, but it returns the same result over some proper subset G' of G only iff there exists a homomorphism from G to G', hence iff the graph is not lean.

As deciding whether a graph is lean is coNP-complete problem [15], the NP-hardness follows immediately. □

Lemma C.8. *The problem* MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$*, when both,* $\mathcal{R}$ *and* $\mathcal{C}$ *are fixed but* $\mathcal{Q}$ *may contain arbitrary CQs is* Σ_3^P*-hard, already if* $\mathcal{Q}$ *contains a single CQ only. It even remains* Σ_3^P*-hard for the case where* $\mathcal{R}=\emptyset$*.*

Proof. We prove the lemma by reduction from the well-known Σ_3^P-complete problem QSAT$_3$. Therefore, let an arbitrary instance of QSAT$_3$ be given by $F = \exists\vec{x}_1\forall\vec{y}_1\exists\vec{x}_2\bigwedge_{i=1}^{n} C_i$, where $C_i = (l_{i,1}\vee l_{i,2}\vee l_{i,3})$ (obviously, the restriction to 3CNF goes w.l.o.g.).

In the following, let $r=|\vec{x}_1|$, $s=|\vec{y}_1|$, $t=|\vec{x}_2|$, and $V=\vec{x}_1\cup\vec{y}_1\cup\vec{x}_2$, where for $v_i\in V$, $v_i\in$
$$\begin{cases}\vec{x}_1 & \text{if } i\leq r\\ \vec{y}_1 & \text{if } r<i\leq r+s\\ \vec{x}_2 & \text{if } r+s<i\leq r+s+t\end{cases}$$
By slight abuse of notation, we use v_i to denote both, URIs in the instance of MINI-RDF$^{\subseteq,CQ}$ and variables in F. Given F, we define G as follows: (Note that for the sake of readability, we define the reduction first in terms of ternary relations, namely from the schema $S=\{V_1/3, D/2, C/3, U_1/1, U_2/1\}$, and translate it afterwards to RDF.)
Let $\bar{G}=\bar{G}_1\cup\bar{G}_2\cup\bar{G}_{u1}\cup\cup\bar{G}_{01}$ with
$\bar{G}_1=\{V_1(v_i,0,1), V_1(v_i,1,0)\mid v_i\in\vec{x}_1\}$,
$\bar{G}_2=\{D(0,1), D(1,0)\}$,
$\bar{G}_{u1}=\{U_1(v_i)\mid v_i\in\vec{x}_1\}$ (where v_i is a new URI for every $v_i\in V$),
$\bar{G}_{01}=\{C(0,0,0), C(0,0,1), C(0,1,0), C(1,0,0), C(0,1,1), C(1,0,1), C(1,1,0), C(1,1,1)\}$.
Then let $\bar{\mathcal{C}}=$
$\big\{\{C(0,0,0)\}\Rightarrow\{t\}\mid \text{for every } t\in G_{01}\big\}\cup$
$\big\{\{C(0,0,0), U_1(V)\}\Rightarrow\{V_1(V,0,1), V_1(V,1,0)\}\big\}\cup$
$\big\{\{C(0,0,0)\}\Rightarrow\bar{G}_2\big\}\cup$
$\big\{\{V_1(V,1,0), V_1(V,0,1)\}\Rightarrow\{C(0,0,0)\}\big\}$
and finally let $\bar{\mathcal{Q}}$ contain the single query
$\bar{q}=\bigwedge_{i=1}^{r} V_1(v_i,X_i,\bar{X}_i)\wedge\bigwedge_{i=1}^{s} D(Y_i,\bar{Y}_i)\wedge\bigwedge_{i=1}^{t} D(Z_i,\bar{Z}_i)\wedge$
$\bigwedge_{i=1}^{n} C(L^*_{i,1},L^*_{i,2},L^*_{i,3})\wedge U_1(V_1)$
$\rightarrow ans(V_1, Y_1,\ldots,Y_s)$

where, $L^*_{i,j} = \begin{cases} X_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ \bar{X}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ Y_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ \bar{Y}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ Z_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \\ \bar{Z}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \end{cases}$

Towards the proof of the correctness of this reduction, we note the following:

- Obviously, $q(\bar{G})$ returns all possible combinations of 0 and 1 of length s
- $\bar{G}'$, being a proper subgraph, must not containt $C(0,0,0)$ (as otherwise, G' must contain every tuple from $\bar{G}$).
- For every $v_i \in \vec{x}_1$, $\bar{G}'$, being a proper subgraph, must not containt both atoms $V_1(v_i,1,0)$ and $V_1(v_i,0,1)$ (otherwise the tgd $\{V_1(V,1,0), V_1(V,0,1)\} \Rightarrow \{C(0,0,0)\}$ would be violated).
- For every $v_i \in \vec{x}_1$, $\bar{G}'$ must contain at least one of the two atoms $V_1(v_i,1,0)$ and $V_1(v_i,0,1)$, otherwise $q(\bar{G}') = \emptyset$.
- For every $v_i \in \vec{x}_1$, G' must contain $U_1(v_i)$ as otherwise the corresponding answer with v_i on first or second position cannot be created any more over $\bar{G}'$.
- Assuming F to be satisfied, and given a truth assignment $T = T_1 \cup T_2 \cup T_3$, define $\bar{G}'$ to contain only those tuples from $V_1^{\bar{G}}$ that correspond to T_1.
  Hence the first group of conjuncts in q can be mapped. Now for every homomorphism from the second group to $\bar{G}'$, we can define a corresponding truth assignment on $\vec{y}_1$, for which there must exists an extension to $\vec{x}_2$, which can be retranslated to an extension of the homomorphism from q to $\bar{G}'$. Because the resulting truth assignment satisfies $\bar{F}$, also the last group of conjuncts can be mapped to $\bar{G}'$.
- For the other direction, assume that there exists such a $\bar{G}'$. From this, define the corresponding truth assingment on $\vec{x}_1$. Now for every truth assignment on $\vec{y}_1$, there exists a corresponding mapping from the Y_i's to $\{0,1\}$, for which there must exists an extension to the Z_i's, such that the whole query body is mapped to $\bar{G}'$. This can be retranslated to a truth assignment on $\vec{x}_2$ such that the overall truth assignment satisfies $\bar{F}$, as it maps in every conjunct at least one literal to $true$.

Using the same ideas as in the previous sections, translating this reduction into RDF syntax, we derive at the following instance:
The RDF graph G is defined as $G = G_1 \cup G_2 \cup G_p \cup G_{u1} \cup G_{01} \cup G_{000}$ where
$G_1 = \{v_i\ r_1\ a_{01}\ .\ v_i\ r_1\ a_{10} \mid v_i \in \vec{x}_1\}$
$G_2 = \{0\ d\ 1\ .\ 1\ d\ 0\}$
$G_p = \{1\ p_1\ a_{10}\ .\ 0\ p_1\ a_{01}\ .\ 0\ p_2\ a_{10}\ .\ 1\ p_2\ a_{01}\}$
$G_{u1} = \{v_i\ u_1\ v_i \mid v_i \in \vec{x}_1\}$
$G_{01} = \{\ 0\ h_1\ a_{001}\ .\ 0\ h_2\ a_{001}\ .\ 1\ h_3\ a_{001}\ .\ 0\ h_1\ a_{010}\ .\ 1\ h_2\ a_{010}\ .\ 0\ h_3\ a_{010}\ .$
$1\ h_1\ a_{100}\ .\ 0\ h_2\ a_{100}\ .\ 0\ h_3\ a_{100}\ .\ 0\ h_1\ a_{011}\ .\ 1\ h_2\ a_{011}\ .\ 1\ h_3\ a_{011}\ .$
$1\ h_1\ a_{101}\ .\ 0\ h_2\ a_{101}\ .\ 1\ h_3\ a_{101}\ .\ 1\ h_1\ a_{110}\ .\ 1\ h_2\ a_{110}\ .\ 0\ h_3\ a_{110}\ .$

$1\ h_1\ a_{111}\ .\ 1\ h_2\ a_{111}\ .\ 1\ h_3\ a_{111}\}$
$G_{000} = \{0\ h_1\ a_{000}\ .\ 0\ h_2\ a_{000}\ .\ 1\ h_3\ a_{000}\}$
Then $\bar{\mathcal{C}}$ translates to
$\mathcal{C} = \{G_{000} \Rightarrow G_{01}\} \cup$
$\{G_{000} \Rightarrow G_2\} \cup$
$\{G_{000} \Rightarrow G_p\} \cup$
$\{G_{000} \cup \{V\ u_1\ V\} \Rightarrow \{V\ r_1\ a_{01}\ .\ V\ r_1\ a_{10}\}\} \cup$
$\{\{V\ r_1\ a_{01}\ .\ V\ r_1\ a_{10}\} \Rightarrow G_{000}\}$
and finally, $\mathcal{Q}$ contains
$q = \{G_q \rightarrow ans(V_1, Y_1, \ldots, Y_s)\}$ where
$G_q = \{v_i\ r_1\ J_i\ .\ X_i\ p_1\ J_i\ .\ \bar{X}_i\ p_2\ J_i \mid i \in [r]\} \cup$
$\{Y_i\ d\ \bar{Y}_i \mid i \in [s]\} \cup$
$\{Z_i\ d\ \bar{Z}_i \mid i \in [t]\} \cup$
$\{L^*_{i,1}\ h_1\ R_i\ .\ L^*_{i,2}\ h_2\ R_i\ .\ L^*_{i,3}\ h_3\ R_i \mid i \in [n]\} \cup$
$\{V_1\ u_1\ V_1\}$
where we introduce new blank nodes $X_i, \bar{X}_i, Y_i, \bar{Y}_i, Z_i, \bar{Z}_i, J_i, R_i, V_1$, and

$$L^*_{i,j} = \begin{cases} X_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ \bar{X}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ Y_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ \bar{Y}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ Z_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \\ \bar{Z}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \end{cases}$$

It remains to show that this reduction is indeed correct, i.e. that F is satisfiable iff there exists a subgraph $G' \subset G$, s.t. (1) G' satisfies $\mathcal{C}$ and (2) $q(G) = q(G')$. To do so, we need the following claim:

$q(G) = \{v_i \mid v_i \in \vec{x}_1\} \times \{0,1\}^s$

To see that this holds, first note that $q(G) \subseteq \{v_i \mid v_i \in \vec{x}_1\} \times \{0,1\}^s$ is trivially true. Just note that all triples with u_1 on predicate position contain only v_i for $i \in [r]$ on subject and object position. Similar, triples with h_1, h_2 and h_3 on predicate position only contain 0 and 1 on subject position, which restricts the domain for V_1 to $\{v_i \mid v_i \in \vec{x}_1\}$ and the domain for every Y_i ($i \in [s]$) to $\{0,1\}$. Now the above expression describes exactly all tuples that can be created using this domain. Hence $q(G)$ cannot contain more tuples.
To see that also $\{v_i \mid v_i \in \vec{x}_1\} \times \{0,1\}^s \subseteq q(G)$ holds, take an arbitrary tuple t from $\{v_i \mid v_i \in \vec{x}_1\} \times \{0,1\}^s$. (Denote with $t[V_1]$ the value in t of V_1, and for all other blank nodes accordingly.) We now define a homomorphism $\mu\colon q \rightarrow G$. Let $\mu(V_1) = t[V_1]$ and $\mu(Y_i) = t[Y_i]$ ($i \in s$). Obviously, $\{\mu(V_1)\ u_1\ \mu(V_1)\} \subseteq G$. Choosing $\mu(X_i)$ ($i \in [r]$) and $\mu(Z_i)$ ($i \in [t]$) arbitrarily from $\{0,1\}$, and defining $\mu(\bar{X}_i) = 1 - \mu(X_i)$ ($i \in [r]$), $\mu(\bar{Y}_i) = 1 - \mu(Y_i)$ ($i \in [s]$), $\mu(\bar{Z}_i) = 1 - \mu(Z_i)$ ($i \in [t]$), and $\mu(J_i) = a_{\mu(X_i)\mu(\bar{X}_i)}$ (for $i \in [r]$), it is easy to check that also $\{v_i\ r_1\ \mu(J_i)\ .\ \mu(X_i)\ p_1\ \mu(J_i)\ .\ \mu(\bar{X}_i)\ p_2\ \mu(J_i) \mid i \in [r]\} \subseteq G$, $\{\mu(Y_i)\ d\ \mu(\bar{Y}_i) \mid i \in [s]\} \subseteq G$, and $\{\mu(Z_i)\ d\ \mu(\bar{Z}_i) \mid i \in [t]\} \subseteq G$. Now it is easy to see that also (for every $i \in [n]$) for $\{\mu(L^*_{i,1})\ h_1\ R_i\ .\ \mu(L^*_{i,2})\ h_2\ R_i\ .\ \mu(L^*_{i,3})\ h_3\ R_i\}$ there exists a value for $\mu(R_i)$, s.t. all three triples are mapped to G, just take $\mu(R_i) = a_{\mu(L^*_{i,1})\mu(L^*_{i,2})\mu(L^*_{i,3})}$.

Now we show both directions of the "if and only if" statement separately:

"If" direction) Assume that there exists a subgraph G' that satisfies (1) and (2). We have to show that then F is satisfiable. We start with some obversations about properties every proper subgraph G' of G that satisfies (1) and (2) must have:

$G_{u1} \subseteq G'$: Assume to the contrary that some triple from G_{u1} is not in G', say $v_{\hat{i}}\ u_1\ v_{\hat{i}}$. But then $q(G')$ cannot contain any tuple t with $t[V_1] = v_{\hat{i}}$, which is a contradiction to (2).
$G_2 \subseteq G'$: Again assume that this is not the case, w.l.o.g. assume that G' does not contain $0\ d\ 1$. It follows immediately that then $q(G')$ cannot contain any tuple t with $t[Y_i] = 0$ for some $i \in [s]$, which is again a contradition to (2).
$G_{000} \not\subseteq G'$: Assume that $G_{0000} \subseteq G'$. It is easy to verify that then $\mathcal{C}$ is only satisfied by G' if $G' = G$, which is a contradiction to the assumption that $G' \subset G$. For every $i \in [r]$: $\{v_i\ r_1\ a_{01}\,.\,v_i\ r_1\ a_{10}\} \not\subseteq G'$: If this would be the case for some $i \in [r]$, then by the last tgd, G_{000} had to be contained in G', which we showed above is not allowed.
For every $i \in [r]$: either $v_i\ r_1\ a_{01}$ or $v_i\ r_1\ a_{10}$ must be contained in G': Assume that this does not hold for some $\hat{i} \in [r]$. Then the corresponding triple $v_{\hat{i}}\ r_1\ j_{\hat{i}}$ in G_q could not be mapped to G', hence $q(G')$ returns no result, which contradicts assumption (2).

Based on these observations, we now define a truth assignment T_1 on $\vec{x}_1$ as follows: For every $i \in [r]$, if $v_i\ r_1\ a_{10} \in G'$, then $T_1(v_i) = \mathit{true}$. If on the other hand $v_i\ r_1\ a_{01} \in G'$, then $T_1(v_i) = \mathit{false}$. Due to the observations made above, T_1 is a well defined truth assignment. Now let T_2 be an arbitrary extension of T_1 to $\vec{y}_1$. From this, we define a mapping $\mu\colon V_i \to \{0,1\}$ on the blank nodes V_i ($i \in [s]$) as follows: For every $i \in [s]$, if $T_2(v_{r+i}) = \mathit{true}$, then $\mu(V_i) = 1$, and if $T_2(v_{r+i}) = \mathit{false}$, then $\mu(V_i) = 0$. By extending μ to V_1 by choosing $\mu(V_1)$ arbitrarily from $\{v_i \mid i \in [r]\}$, we know from (2) that $(\mu(V_1), \mu(Y_1), \ldots, \mu(Y_r)) \in q(G')$. Therefore there exists an extension μ' of μ that maps G_q into G' (i.e. μ' is a homomorphism from G_q to G'). Moreover, it is easy to check that for every $i \in [t]$, $\mu'(Z_i) = 1 - \mu'(\bar{Z}_i)$. Therefore, the truth assignment T_3 on $\vec{x}_2$ defined by $T_3(v_{r+s+i}) = \mathit{true}$ if $\mu'(Z_i) = 1$ and $T_3(v_{r+s+i}) = \mathit{false}$ if $\mu'(\bar{Z}_i) = 1$ (for $i \in [t]$) is well defined.

It remains to show that under the truth assignment $T = T_1 \cup T_2 \cup T_3$ indeed every clause C_i in F evaluates to true. To see this, note that for no $i \in [n]$, $\mu'(L_{i,1}) = \mu'(L_{i,2}) = \mu'(L_{i,3}) = 0$, which holds because G_{000} is not contained in G', and therefore there exists no value for $\mu'(R_i)$ s.t. there are three triples in G' with h_1, h_2, and h_3 on predicate position, the same value $\mu'(R_i)$ on object position and 0 on subject position. Hence for every $i \in [n]$, at least one $L^*_{i,j}$ is mapped to one. But then the corresponding literal $l_{i,j}$ in F evaluates to true under T: If $L^*_{i,j}$ is $\bar{X}_\alpha$ (resp. $\bar{Y}_\alpha$ or $\bar{Z}_\alpha$), then $l_{i,j}$ is of the form $\neg v_\alpha$ (resp. $\neg v_{\alpha+r}$ or $\neg v_{\alpha+r+s}$). But then, by definition, $T(v_\alpha) = \mathit{false}$ (resp. $T(v_{\alpha+r}) = \mathit{false}$ or $T(v_{\alpha+r+s}) = \mathit{false}$), and therefore $T(l_{i,j}) = \mathit{true}$. On the other hand, if $L^*_{i,j}$ is X_α (resp. Y_α or Z_α), then $l_{i,j}$ is of the form v_α (resp. $v_{\alpha+r}$ or $v_{\alpha+r+s}$). But then, by definition, $T(v_\alpha) = \mathit{true}$ (resp. $T(v_{\alpha+r}) = \mathit{true}$ or $T(v_{\alpha+r+s}) = \mathit{true}$), and therefore $T(l_{i,j}) = \mathit{true}$, which proves the case.

"Only if" direction) We assume that F is satisfiable, and have to show that there exists a proper subgraph $G' \subset G$ that satisfies (1) and (2). As F is satisfiable, we know that there exists a truth assignment T_1 on $\vec{x}_1$, s.t. for all truth assignments T_2 on $\vec{y}_1$ there exists a truth assignment T_3 on $\vec{x}_2$, s.t. every C_i ($i \in [n]$) evaluates to true under $T = T_1 \cup T_2 \cup T_3$.

Given T_1, we define G' as follows: $G' = G \setminus (G_{000} \cup \{\tau_i \mid i \in [r]\})$, where $\tau_i = v_i\ r_1\ a_{01}$ if $T(v_i) = \mathit{true}$ and $\tau_i = v_i\ r_1\ a_{10}$ if $T(v_i) = \mathit{false}$. (Recall that v_i denotes both, URIs in G and variables in F.)

It is easy to see that G' satisfies $\mathcal{C}$: The first four tgds are satisfied because $G_{000} \not\subseteq G'$, hence the antecedent is not satisfied, and the last tgd is satisfied because T_1 assigns exactly one truth value to every $v_i \in \vec{x}_1$, hence for every $i \in [r]$, exactly one of the two triples $v_i\ r_1\ a_{10}\,.\,v_i\ r_1\ a_{01}$ is not in G', therefore again the antecedent of the tgd is not satisfied. From this, it follows trivially that G' is a proper subset of G. It therefore only remains to show, that indeed $q(G') = q(G)$.

That is, given an arbitrary tuple $t \in q(G)$, we have to show that there exists an extension μ' of the mapping μ defined by $\mu(V_1) = t[V_1]$ and $\mu(Y_i) = t[Y_i]$ (for every $i \in [s]$), s.t. $\mu'(body(q)) \subseteq G'$. Towards this goal, we first define a truth asisssgnment T_2 on $\vec{y}_1$ as follows: For $v_i \in \vec{y}_1$, $T_2(v_i) = true$ if $\mu(Y_i) = 1$, and $T_2(v_i) = false$ if($\mu(Y_i) = 0$). We know by assumption that now there must exist some truth assignment T_3 on $\vec{x}_2$, s.t. every C_i ($i \in [n]$) evaluates to $true$ under $T = T_1 \cup T_2 \cup T_3$.

We now define an extension μ' of μ (i.e. a homomorphism on the blank nodes not already mapped by μ) as follows:
$\mu'(X_i) = 1$ if $T(v_i) = true$ (resp. $\mu'(X_i) = 0$ if $T(v_i) = false$), and $\mu'(\bar{X}_i) = 1 - \mu'(X_i)$ (for $i \in [r]$).
$\mu'(\bar{Y}_i) = 1 - \mu'(Y_i)$ (for $i \in [s]$).
$\mu'(Z_i) = 1$ if $T(v_i) = true$ (resp. $\mu'(Z_i) = 0$ if $T(v_i) = false$), and $\mu'(\bar{Z}_i) = 1 - \mu'(Z_i)$ (for $i \in [t]$).
It is easy to see that $\mu'(V_1)\ u_1\ \mu'(V_1) \in G'$, as all triples in G with u_1 on predicate position are also in G'. Further, also all triples $\mu'(Y_i)\ d\ \mu'(\bar{Y}_i)$ (for $i \in [s]$) are in G' (resp. all triples $\mu'(Z_i)\ d\ \mu'(\bar{Z}_i)$ for $i \in [t]$). Just note that G' still contains both, $0\ d\ 1$ and $1\ d\ 0$, and that μ' maps every Y_i and $\bar{Y}_i$ (resp. Z_i and $\bar{Z}_i$) to different values. To further see that also $v_i\ r_1\ J_i\,.\,X_i\ p_1\ J_i\,.\,\bar{X}_i\ p_2\ J_i$ (for every $i \in [r]$) can be mapped to G', consider the extension of μ' by $\mu'(J_i) = a_{\mu'(X_i)\mu'(\bar{X}_i)}$. Finally, if remains to show that also the triples $L^*_{i,1}\ h_1\ R_i\,.\,L^*_{i,2}\ h_2\ R_i\,.\,L^*_{i,3}\ h_3\ R_i$ can be mapped to G' (for $i \in [n]$). From the fact that under T, every C_i evaluates to $true$, and the construction of μ' from T, we can derive that for $i \in [n]$, not all three of $L^*_{i,1}$, $L^*_{i,2}$, and $L^*_{i,3}$ are mapped to 0 by μ'. But for all other possibilities of assigning 0 and 1 to $L^*_{i,1}$, $L^*_{i,2}$, and $L^*_{i,3}$, the triples can be mapped to G', just consider $\mu'(R_i) = a_{\mu(L^*_{i,1})\mu(L^*_{i,2})\mu(L^*_{i,3})}$, which proves the case. □

Lemma C.9. *The problem* MINI-RDF$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$*, when $\mathcal{R}$ is fixed, $\mathcal{C} = \emptyset$, and $\mathcal{Q}$ may contain arbitrary CQs is Π^P_2-hard, already if $\mathcal{Q}$ contains a single CQ only. The problem remains Π^P_2-hard, even if $\mathcal{R} = \emptyset$.*

Proof. We prove the lemma by reduction from the well-known Π^P_2-complete problem coQSAT$_2$. Therefore, let an arbitrary instance of coQSAT$_2$ be given by $F = \forall\vec{x}_1 \exists\vec{y}_1 \bigwedge_{i=1}^n C_i$, where $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ (obviously, the restriction to 3CNF is w.l.o.g.). In the following, let $r = |\vec{x}_1|$ and $s = |\vec{y}_1|$. We now define G as follows (Note that for the sake of readability, we first describe the reduction in terms of ternary relations, namely for the schema $S = \{Q/2, C/3, D/3\}$, and show afterwards its translation to an RDF graph):
Let $\bar{G} = \{Q(0,1), Q(1,0), C(0,0,0), C(0,0,1), D(0,0,1),$
$C(0,1,0), D(0,1,0), C(1,0,0), D(1,0,0), C(0,1,1), D(0,1,1),$
$C(1,0,1), D(1,0,1), C(1,1,0), D(1,1,0), C(1,1,1), D(1,1,1)\}$
and let $\bar{\mathcal{Q}}$ contain the single query
$\bar{q} = \bigwedge_{i=1}^n C(L^*_{i,1}, L^*_{i,2}, L^*_{i,3}) \wedge \bigwedge_{i=1}^r Q(Y_i, \bar{Y}_i) \wedge \bigwedge_{i=1}^s Q(Z_i, \bar{Z}_i) \wedge$
$C(V_1, V_2, V_3) \wedge D(V_1, V_2, V_3)$
$\rightarrow ans(V_1, V_2, V_3, Y_1, \ldots, Y_r)$,
where $L^*_{i,j} = \begin{cases} Y_\beta & \text{if } l_{i,j} = x_\alpha \text{ and } x_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ \bar{Y}_\beta & \text{if } l_{i,j} = \neg x_\alpha \text{ and } x_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ Z_\beta & \text{if } l_{i,j} = y_\alpha \text{ and } y_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ \bar{Z}_\beta & \text{if } l_{i,j} = \neg y_\alpha \text{ and } y_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \end{cases}$

We shortly sketch the intuition behind this reduction:

- First of all, it is quite easy to see that over $\bar{G}$, q returns all possibilities how to combine the values in D (i.e. $(0,0,1), \ldots, (1,1,1)$) with all possible combinations of 0 and 1 of length r.

- If removing any entry from $\bar{G}$ except $C(0,0,0)$, it is immediate that certain types of results in $q(\bar{G})$ cannot be created any more over $\bar{G}'$. (Either all Y_i are bound to the same value, or one of $(0,0,1),\ldots,(1,1,1)$ for (V_1,V_2,V_3) can no longer be created.)

- Now consider $\bar{G}' = \bar{G}\setminus\{C(0,0,0)\}$, and $q(\bar{G}') = q(\bar{G})$. Then every homomorphism from $body(q)$ to $\bar{G}'$ encodes a satisfying truth assignment for $\bigwedge_{i=1}^{n} C_i$. (For no clause C_i, all three literals are mapped to $false$.)

- On the other hand, if F is satisfied, than every satisfying truth assignment on $\bigwedge_{i=1}^{n} C_i$ gives rise to homomorphism $\mu\colon body(q) \rightarrow \bar{G}'$, s.t. $\mu(head(q))$ uniquely describes the truth assignment on $\vec{x}_1$. As $\bigwedge_{i=1}^{n} C_i$ is satisfiable for every truth assignment on $\vec{x}_1$, the same results as for $q(\bar{G})$ are created.

When translated into RDF syntax, we derive the following RDF graph $G = G^1_{01} \cup G^2_{01} \cup G_{000} \cup G_d$ where

$$
\begin{aligned}
G^1_{01} = \{\ & 0\ h^1_1\ a_{001}\ .\ 0\ h^1_2\ a_{001}\ .\ 1\ h^1_3\ a_{001}\ .\ 0\ h^1_1\ a_{010}\ .\ 1\ h^1_2\ a_{010}\ .\ 0\ h^1_3\ a_{010}\ .\\
& 1\ h^1_1\ a_{100}\ .\ 0\ h^1_2\ a_{100}\ .\ 0\ h^1_3\ a_{100}\ .\ 0\ h^1_1\ a_{011}\ .\ 1\ h^1_2\ a_{011}\ .\ 1\ h^1_3\ a_{011}\ .\\
& 1\ h^1_1\ a_{101}\ .\ 0\ h^1_2\ a_{101}\ .\ 1\ h^1_3\ a_{101}\ .\ 1\ h^1_1\ a_{110}\ .\ 1\ h^1_2\ a_{110}\ .\ 0\ h^1_3\ a_{110}\ .\\
& 1\ h^1_1\ a_{111}\ .\ 1\ h^1_2\ a_{111}\ .\ 1\ h^1_3\ a_{111}\ \}\\
G^2_{01} = \{\ & 0\ h^2_1\ a_{001}\ .\ 0\ h^2_2\ a_{001}\ .\ 1\ h^2_3\ a_{001}\ .\ 0\ h^2_1\ a_{010}\ .\ 1\ h^2_2\ a_{010}\ .\ 0\ h^2_3\ a_{010}\ .\\
& 1\ h^2_1\ a_{100}\ .\ 0\ h^2_2\ a_{100}\ .\ 0\ h^2_3\ a_{100}\ .\ 0\ h^2_1\ a_{011}\ .\ 1\ h^2_2\ a_{011}\ .\ 1\ h^2_3\ a_{011}\ .\\
& 1\ h^2_1\ a_{101}\ .\ 0\ h^2_2\ a_{101}\ .\ 1\ h^2_3\ a_{101}\ .\ 1\ h^2_1\ a_{110}\ .\ 1\ h^2_2\ a_{110}\ .\ 0\ h^2_3\ a_{110}\ .\\
& 1\ h^2_1\ a_{111}\ .\ 1\ h^2_2\ a_{111}\ .\ 1\ h^2_3\ a_{111}\ \}\\
G_{000} = \{ & 0\ h^1_1\ a_{000}\ .\ 0\ h^1_2\ a_{000}\ .\ 1\ h^1_3\ a_{000}\}\\
G_d = \{ & 0\ d\ 1\ .\ 1\ d\ 0\}
\end{aligned}
$$

which results in $\mathcal{Q}$ containing the query

$q = G_q \rightarrow ans(V_1, V_2, V_3, Y_1, \ldots, Y_r)$ where

$$
\begin{aligned}
G_q = \{ & Y_i\ d\ \bar{Y}_i \mid i \in [r]\} \cup\\
& \{Z_i\ d\ \bar{Z}_i \mid i \in [s]\} \cup\\
& \{L^*_{i,1}\ h^1_1\ R_i\ .\ L^*_{i,2}\ h^1_2\ R_i\ .\ L^*_{i,3}\ h^1_3\ R_i \mid i \in [n]\} \cup\\
& \{V_1\ h^1_1\ J_1\ .\ V_2\ h^1_2\ J_1\ .\ V_3\ h^1_3\ J_1\ .\ V_1\ h^2_1\ J_1\ .\ V_2\ h^2_2\ J_1\ .\ V_3\ h^2_3\ J_1\}
\end{aligned}
$$

It remains to show that this reduction is indeed correct. To see this, consider the following important claim:

$q(G) = \{\{0,1\}^3 \setminus \{(0,0,0)\}\} \times \{0,1\}^s$

To see that $q(G) \subseteq \{\{0,1\}^3 \setminus \{(0,0,0)\}\} \times \{0,1\}^s$, just note that all triples in G with d on predicate position, only have 0 or 1 on subject or object position. Therefore the domain for all Y_i is $\{0,1\}$. Also, all triples with h^1_1, h^1_2, h^1_3, h^2_1, h^2_2, and h^2_3 on predicate position contain only 0 or 1 on subject position, which also restricts the domain of V_1, V_2, and V_3 to $\{0,1\}$. However, there are no three triples with h^2_1, h^2_2, and h^3_2 on predicate position such that V_1, V_2 and V_3 can be all mapped to 0, which is the reason why $(0,0,0)$ is excluded.

To see $\{\{0,1\}^3 \setminus \{(0,0,0)\}\} \times \{0,1\}^s \subseteq q(G)$, consider an arbitrary such tuple t, and the mapping μ defined by it on V_1, V_2, V_3 and all Y_i (i.e. $\mu(V_1) = t[V_1]$, $\mu(V_2) = t[V_2]$, $\mu(V_3) = t[V_3]$, and $\mu(Y_1) = t[Y_i]$ for every $i \in [s]$). We have to show that μ can be extended to a homomorphism $\mu'\colon body(q) \rightarrow G$. Therefore, define $\mu(\bar{Y}_i) = \mu(Y_i)$ (for every $i \in [r]$). Further, pick $\mu(Z_i)$ arbitrarily from $\{0,1\}$ (for $i \in [s]$), and define $\mu(\bar{Z}_i) = 1 - \mu(Z_i)$. Then $Y_i\ d\ \bar{Y}_i$ ($i \in [r]$) and $Z_i\ d\ \bar{Z}_i$ ($i \in [s]$) are in G. By further defining $\mu(J_1) = a_{\mu(V_1)\mu(V_2)\mu(V_3)}$ and the fact that not all three, $\mu(V_1)$, $\mu(V_2)$, and $\mu(V_3)$ are 0, it is also easy to see that $\{\mu(V_1)\ h^1_1\ \mu(J_1)\ .\ \mu(V_2)\ h^1_2\ \mu(J_1)\ .\ \mu(V_3)\ h^1_3\ \mu(J_1)\ .\ \mu(V_1)\ h^2_1\ \mu(J_1)\ .\ \mu(V_2)\ h^2_2\ \mu(J_1)\ .\ \mu(V_3)\ h^2_3\ \mu(J_1)\} \subseteq G$. Finally, by the same argument, μ can be further extended to also map the

remaining triples in $body(q)$, $\{L^*_{i,1}\ h^1_1\ R_i\ .\ L^*_{i,2}\ h^1_2\ R_i\ .\ L^*_{i,3}\ h^1_3\ R_i \mid i \in [n]\}$. By just defining $\mu(R_i) = a_{\mu(L^*_{i,1})\mu(L^*_{i,2})\mu(L^*_{i,3})}$, it is easy to see that for every value of $\mu(R_i)$, G contains the correpsonding triples.

Finally we show that F is satisfiable iff there exists a subgraph $G' \subset G$ s.t. $q(G') = q(G)$. Therefore, we show both directions separately.

"If"-direction: We assume that there exists some $G' \subset G$ s.t. $q(G') = q(G)$. It is easy to check that this only possible if $G' = G \setminus G_{000}$: If G' misses one triple from G_d, then all answers to q over G' can return either only 1, or only 0 for all Y_i ($i \in [r]$), if G' misses both triples from G_d, then $q(G') = \emptyset$. On the other hand, if G' misses some triples from G^1_{01} or G^2_{01}, then at least one of the answers $(0,0,1), \ldots, (1,1,1)$ for (V_1, V_2, V_3) cannot be derived over G'.

Now let an arbitrary truth assignment T_1 be given on $\vec{x}_1$. We have to show that there exists a truth assignment T_2 on $\vec{y}_1$ s.t. $\bar{F} = \bigwedge_{i=1}^n$ evaluates to true under $T = T_1 \cup T_2$. Towards this goal, consider some result from $q(G')$ where $Y_i = 1$ if $T_1(x_i) = \mathit{true}$ and $Y_i = 0$ if $T_1(x_i) = \mathit{false}$. By assumption, there exist exactly seven such results in $q(G)$ (however, in the following we consider only the projection onto $Y_1, \ldots, Y_r$, and also do not consider the triples $\{V_1\ h^1_1\ J_1\ .\ V_2\ h^1_2\ J_1\ .\ V_3\ h^1_3\ J_1\ .\ V_1\ h^2_1\ J_1\ .\ V_2\ h^2_2\ J_1\ .\ V_3\ h^2_3\ J_1\}$ from $body(q)$). From this it follows that there exists a homomorphism $\mu(body(q)) \to G'$ s.t. $\mu(head(q))$ gives the result defined above. We note that for every $i \in [r]$ it holds that $\mu(Y_i) = 1 - \mu(\bar{Y}_i)$, and for every $i \in [s]$ it holds that $\mu(Z_i) = 1 - \mu(\bar{Z}_i)$.

From this homomorphism, we define a truth assignment T_2 on $\vec{y}_1$ as follows. $T_2(y_i) = \mathit{true}$ if $\mu(Z_i) = 1$, and $T_2(y_i) = \mathit{false}$ if $\mu(Z_i) = 1$. We claim that then $\bar{F}$ evaluates to true under $T = T_1 \cup T_2$. First of all, it is easy to see that T_2 is indeed a satisfying truth assignment. μ maps every Z_i ($i \in [s]$) to either 0 or 1, as otherwise there would exists some triple $\tau\colon Z_j\ d\ \bar{Z}_j$ in $body(q)$ s.t. $\mu(\tau) \notin G$, which is a contradiction to the assumption that μ is a homomorphism. On the other hand, as μ is a well defined function, every Z_i is also mapped to at most one value.

It remains to show that T indeed satisfies $\bar{F}$. To see this, note that for no $i \in [n]$, $\mu(L^*_{i,1}) = \mu(L^*_{i,2})\mu(L^*_{i,3}) = 0$, as there does not exists a value for $\mu(R_i)$ s.t. μ would indeed be a valid homomorphism. Hence at least one of $\mu(L^*_{i,1})$, $\mu(L^*_{i,2})$, and $\mu(L^*_{i,3})$ must be mapped to 1 (say $L^*_{i,j}$). But by the definition of T_2 and q, this means that T assigns true to the corresponding literal in C_i: If $L^*_{i,j}$ is some $\bar{Y}_\alpha$ or $\bar{Z}_\alpha$, then by definition $l_{i,j}$ is a negated variable. Hence, if $L^*_{i,j} = 1$, then T assigns false to the variable, and $l_{i,j}$ evaluates to true. If on the other hand, $L^*_{i,j}$ is some Y_α or Z_α, then $l_{i,j}$ is a unnegated variable, and from $L^*_{i,j} = 1$ it follows immediately that T assigns true to it.

"Only if" direction Assume that F is satisfied, i.e. that for every truth assignment T_1 on $\vec{x}_1$, there exists a truth assignment T_2 on $\vec{y}_1$ s.t. $\bar{F}$ evaluates to true under $T = T_1 \cup T_2$. Then we define $G' = G \setminus G_{000}$. We have to show that $q(G') = q(G)$.

First of all, it is easy to see that for V_1, V_2, and V_3, q returns the same results over G' as over G. As these variables are also independent of the other parts of the query, we can omit them in the further discussion, and concentrate on the Y_i. Now consider an arbitrary result t in $q(G)$ (projected onto $Y_1, \ldots, Y_r$). We define a truth assignment T_1 on $\vec{x}_1$ as follows: For every x_i, $T_1(x_i) = \mathit{true}$ if $t[Y_i] = 1$, and $T_1(x_i) = \mathit{false}$ if $t[Y_i] = 0$. It goes without saying that T_1 is a well defined truth assignment. By the assumption we know that there must exists some truth assignmeht T_2, s.t. $T = T_1 \cup T_2$ satisfies $\bar{F}$.

We now define a homomorphism $\mu\colon body(q) \to G'$ from T as follows (again, we consider only the Y_i): For $i \in [r]$, define $\mu(Y_i) = 1$ and $\mu(\bar{Y}_i) = 0$ if $T(x_i) = \mathit{true}$, and $\mu(Y_i) = 0$ and $\mu(\bar{Y}_i) = 1$ if $T(x_i) = \mathit{false}$. For $i \in [s]$, define $\mu(Z_i) = 1$ and $\mu(\bar{Z}_i) = 0$ if $T(y_i) = \mathit{true}$, and $\mu(Z_i) = 0$ and $\mu(\bar{Z}_i) = 1$ if $T(y_i) = \mathit{false}$. Obviously, if μ is indeed a homomorphism, then it creates the answer t.

It remains to show that μ can be extended to some μ', s.t. $\mu'(body(q)) \subseteq G'$. For the triples $Y_i\ d\ \bar{Y}_i$ and $Z_i\ d\ \bar{Z}_i$, this is obviously the case. For the remaining triples, first note that for no clause C_i ($i \in [n]$), all three literals are mapped to $false$ by T. By the definition of q and the construction of μ, this translates to the fact that for every $i \in [n]$, none of the three triples $L^*_{i,1}\ h^1_1\ R_i$, $L^*_{i,2}\ h^1_2\ R_i$, and $L^*_{i,3}\ h^1_3\ R_i$ are mapped to 0 by μ: If $l_{i,j}$ is a unnegated variable, then $L^*_{i,j}$ is either some Y_α or some Z_α, and $\mu(L^*_{i,j}) = 1$, because of $T(l_{i,j}) = true$. If on the other hand, $l_{i,j}$ is a negated variable, then $L^*_{i,j}$ is either some $\bar{Y}_\alpha$ or some $\bar{Z}_\alpha$, and $\mu(L^*_{i,j}) = 1$, because of $T(l_{i,j}) = true$, hence the corresponding variable gets assinged $false$.

But for every possible combinations of 0 and 1 to $L^*_{i,1}$, $L^*_{i,2}$, and $L^*_{i,3}$ different from $(0, 0, 0)$, there exists a corresponding values for $\mu'(R_i)$ s.t. μ' sends the $body(q)$ to G'. Just set $\mu'(R_i) = a_{\mu(L^*_{i,1}L^*_{i,2}L^*_{i,3})}$, which proves the case. □

Considering the results for rule minimisation w.r.t. conjunctive queries, we have to consider the following cases: *Membership* for (II.a), (I.b), and (I.c), as well as hardness for (I.a), (II.b), and (II.c). Concerning the hardness of (I.b), note that both, (I.a) and (II.b) are special cases of this setting.

Lemma C.10. *The problem* RDF-RULEMIN$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{Q})$ *where $\mathcal{R}$ is a set of b-bounded rules and $\mathcal{Q}$ is a set of b-bounded CQs can be decided in* PTIME*. The problem even remains in* PTIME *if $\mathcal{Q}$ is a set of body-b-bounded UCQs.*

Proof. The problem can be decided by the following algorithm:
(1) Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
(2) For every $r \in \mathcal{R}$:
  (a) Set $\mathcal{R}' = \mathcal{R} \setminus \{r\}$
  (b) Compute $\hat{G}' = Cl_{\mathcal{R}'}(G)$
  (c) For every $q \in \mathcal{Q}$, compute $q(\hat{G})$ and $q(\hat{G}')$,
    and check if they are the same.
  (d) If it holds for all $q \in \mathcal{Q}$, return *"Yes"*,
    otherwise try next $r \in \mathcal{R}$
(3) Return *"No"*

Under the restrictions assumed on the input, this alorithm runs in polynomial time: By [11, Proposition 9], $Cl_{\mathcal{R}}(G)$ (and also $Cl_{\mathcal{R}'}(G)$) can be computed in polynomial time. As the size of $Cl_{\mathcal{R}}(G)$ and every $Cl_{\mathcal{R}'}(G)$ is polynomially bounded in the size of the input, and because the body of each $q \in \mathcal{Q}$ is b-bounded, also step (2c) fits into polynomial time, as at most $|AD|^{3 \cdot b}$ homomorphisms need to be checked for every $q \in \mathcal{Q}$ (where AD denotes the active domain).

It is also easy to see that step (2c) still fits into polynomial time if $\mathcal{Q}$ is allowed to contain UCQs. □

Lemma C.11. *The problem* RDF-RULEMIN$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{Q})$*, for arbitrary rules $\mathcal{R}$ and head-b-bounded CQs $\mathcal{Q}$, can be solved in Δ^P_2. The problem remains in Δ^P_2 also for UCQs $\mathcal{Q}$.*

Proof. The following deterministic algorithm, using an NP-oracle, decides the problem and requires only polynomial time:
(1) Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
(2) For every $q \in \mathcal{Q}$, compute (and store) $\hat{q} = q(\hat{G})$.

(3) For every $r \in \mathcal{R}$
(a) Set $\mathcal{R}' = \mathcal{R} \setminus \{r\}$
(b) Compute $\hat{G}' = Cl_{\mathcal{R}'}(G)$
(c) For every $q \in \mathcal{Q}$, compute $q(\hat{G}')$,
and check if $\hat{q} = q(\hat{G}')$
The program returns *"Yes"* if for some $r \in \mathcal{R}$, $\hat{q} = q(\hat{G}')$ holds for all $q \in \mathcal{Q}$, otherwise it returns *"No"*.

To see that this algorithm runs in polynomial time using an NP-oracle, consider the following: Step (1) and (2) fit into Δ_2^P because the number of possible result triples is at most polynomial in the size of the input, and checking for every possible triple whether it can be derived by a rules or query can be done by a call to the NP-oracle (for concrete algorithms see the proof of Theorem 4.2 for step (1) and the proof of Lemma C.3 for step (2)). The timebound for step (3b) holds for the same reason as the one for step (1), just as the timebound for step (3c) follows immediately from the one of step (2).

Allowing for UCQs does obviously not change these complexity results. □

Lemma C.12. *The problem* RDF-RULEMIN$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{Q})$*, for arbitrary $\mathcal{R}$ where $\mathcal{Q}$ may contain arbitrary CQs can be solved in Π_2^P. The problem remains in Π_2^P even if $\mathcal{Q}$ is a set of arbitrary UCQs.*

Proof. To prove this result, we devise a nondeterministic, polynomial algorithm that, using a coNP-oracle, decides the co-problem, i.e. the question whether $\mathcal{R}$ is already minimal. (I.e. we show Σ_2^P membership for the co-problem.)

let $\mathcal{R} = \{r_1, \ldots, r_n\}$
for every $i \in [n]$, let $\mathcal{R}_i = \mathcal{R} \setminus \{r_i\}$
(1) Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
(2) Compute $\hat{G}_i = Cl_{\mathcal{R}_i}(G)$
(3) Guess n (not necessarily distinct) queries $q_i \in \mathcal{Q}$
(4) Guess n homomorphisms $\tau_1, \ldots, \tau_n$, with $\tau_i \colon body(q_i) \to \hat{G}$
(5) Check $\forall i \in [n]$ by a call to a coNP-oracle (i.e. using n calls)
that $\tau_i(head(q_i)) \notin q(\hat{G}_i)$, i.e. check that for all homomorphisms
$\tau_i' \colon body(q_i) \to \hat{G}_i$ with $\tau_i'(body(q_i)) \subseteq \hat{G}_i$, whether
$\tau_i'(head(q_i)) \neq \tau(head(q_i))$
(6) $\mathcal{R}$ is minimal, if all n oracle calls return "*yes*"

The intuition of this is as follows. After computing the closure of G under $\mathcal{R}$ and under all "candidate" rulesets $\mathcal{R}_i$, we guess for every $\mathcal{R}_i$ some query q and an answer over $\hat{G}$ to this query, such that the answer cannot be created by q over $\hat{G}_i$. If this is the case, $\mathcal{R}$ cannot be further minimised.

To see that this algorithm indeed runs in polynomial time, note that computing the closure under (subsets of) $\mathcal{R}$ (steps (1), (2)) fits into Δ_2^P (i.e. into polynomial time with a NP-oracle), and therefore also into Σ_2^P. Guessing the queries and the results of this queries over $\hat{G}$ (steps (3), (4)) introduces the nondeterminism. Checking for each result whether it can not be created over $\hat{G}_i$ can be done in polynomial time by a call to the coNP-oracle. Therefore the overall algorithm requires only polynomial time.

The extension to UCQs is analogously to the proof of Lemma C.4: It suffices to replace step (3) by
(3) For $i \in [n]$, guess $Q_i \in \mathcal{Q}$ and $q_i \in Q_i$
and in step (5) to check that $\tau_i(head(q_i)) \notin Q(\hat{G}_i)$, which means that the coNP-oracle has to check for all $q \in Q_i$ that $q(\hat{G}_i)$ does not create $\tau_i(head(q_i))$. However, this additional check obviously can still be carried out by the oracle. □

Lemma C.13. *The problem* RDF-RULEMIN$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{Q})$ *is* coNP*-hard if $\mathcal{R}$ contains arbitrary rules and $\mathcal{Q}$ is restricted to b-bounded CQs. The problem remains* coNP*-hard, even if each, $\mathcal{R}$ and $\mathcal{Q}$ contain a single element each.*

Proof. The result follows immediately from Theorem 4.2 and the observation that the hardness result carries over: Considering $\mathcal{Q} = \{q\}$, where $q = \{S\ P\ O\} \rightarrow ans(S,P,O)\}$, it is obvious that for some $\mathcal{R}' \subset \mathcal{R}$, $Cl_{\mathcal{R}}(G) \subseteq Cl_{\mathcal{R}'}(G)$ iff $q(Cl_{\mathcal{R}}(G)) \subseteq q(Cl_{\mathcal{R}'}(G))$. As testing whether such an $\mathcal{R}'$ exists is coNP-complete by Theorem 4.2, the coNP-hardness for RDF-RULEMIN$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{Q})$ follows immediately.

To see that also the second part of the theorem holds, just note that $\mathcal{Q}$ contains only a single element (q), and that the reduction given in the proof of Theorem 4.2 requires a single rule only, too. □

Lemma C.14. *The problem* RDF-RULEMIN$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{Q})$ *is* NP*-hard if all rules in $\mathcal{R}$ are b-bounded and $\mathcal{Q}$ contains head-b-bounded CQs. The problem remains* NP*-hard, even if each, $\mathcal{R}$ and $\mathcal{Q}$ contain a single element each.*

Proof. The proof is done by the reduction from the well-known NP-complete problem 3-Colorability. Let an arbitrary instance of co-3-Colorability be given by the graph (V,E) and let $V = \{v_1,\ldots,v_n\}$. Then we construct the RDF graph G, the rule set $\mathcal{R}$, and the set of queries $\mathcal{Q}$ as follows:

$G = \{0\ e\ 1\,.\,0\ e\ 2\,.\,1\ e\ 2\,.\,1\ e\ 0\,.\,2\ e\ 0\,.\,2\ e\ 1\}$

$\mathcal{R} = \{r\}$ with $r = \{X_\alpha\ e\ X_\beta\} \Rightarrow \{0\ e\ 0\}$.

$\mathcal{Q} = \{q\}$ with $q = \{X_\alpha\ e\ X_\beta \mid (v_\alpha, v_\beta) \in E\} \rightarrow ans()$
(where we introduce a new blank node X_i for every $v_i \in V$).

Clearly, this reduction is feasible in polynomial time. For the correctness, we observe that $Cl_{\mathcal{R}}(G) \subseteq G \cup \{0\ e\ 0\}$. Hence $q(Cl_{\mathcal{R}}(G)) \neq \emptyset$, as the homomorphism that maps every blank node in q to 0 trivially maps $body(q)$ to G.

It remains to show that r is redundant iff the given graph is 3-colorable.

First, assume that the graph is 3-colorable. Express a valid 3-coloring by a mapping $\tau\colon\ : V \rightarrow \{0,1,2\}$. Then we define a mapping $\tau'\colon \{X_1,\ldots,X_n\} \rightarrow \{0,1,2\}$ as $\tau'(X_i) = \tau(v_i)$ (where X_i is the blank node corresponding to v_i). By assumption, for no two nodes $v_i, v_j \in V$, connected by a edge, $\tau(v_i) = \tau(v_j)$. But for all other combinations, the triple $\tau'(X_i)\ e\ \tau'(X_j)$ is in G, hence $q(G) \neq \emptyset$, and therefore r is redundant.

On the other hand, assume that r is redundant, i.e. $q(G) \neq \emptyset$. Then there exists a homomorphism $\tau\colon body(q) \rightarrow G$. By definition of q, $body(q)$ contains a triple $X_i\ e\ X_j$ for every edge $(v_i, v_j) \in E$. But because G does not contain a triple with e on predicate position and the same value on subject and object position, all such X_i, X_j are mapped to different values. This immediately gives a valid 3-coloring for the graph. □

Lemma C.15. *The problem* RDF-RULEMIN$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{Q})$*, where $\mathcal{R}$ may contain arbitrary rules and $\mathcal{Q}$ may contain arbitrary CQs, is Π_2^P-hard. The problem remains Π_2^P-hard, even if both, $\mathcal{R}$ and $\mathcal{Q}$ contain a single element only.*

Proof. The proof is done by reduction from the well-known Π_2^P-hard problem $\forall$-QSAT$_2$. Once more, we state the reduction in terms of ternary relations only, the translation into RDF syntax is straight forward. Using the schema $S = \{D/2, C/3\}$, we fix $\bar{G}$ and $\bar{\mathcal{R}}$ as:
$\bar{G} = \{D(0,1), D(1,0)\} \cup$

$\{C(0,0,1), C(0,1,0), C(1,0,0), C(0,1,1), C(1,0,1), C(1,1,0), C(1,1,1)\}$
and
$\bar{\mathcal{R}} = \{\{D(0,1)\} \Rightarrow \{C(0,0,0)\}\}$.

Now let an arbitrary instance of of $\forall$-QSAT$_2$ be given by the formula $F = \forall\vec{x}_1 \exists\vec{y}_1 \bigwedge_{i=1}^{n} C_i$ where $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ (obviously, the restriction to 3CNF is w.l.o.g.). In the following, let $r = |\vec{x}_1|$ and $s = |\vec{y}_1|$. Introducing two new blank nodes X_i and $\bar{X}_i$ for every $x_i \in \vec{x}_1$, and two new blank nodes Y_i and $\bar{Y}_i$ for every $y_i \in \vec{y}_1$, we now define the single query $q \in \bar{\mathcal{Q}}$ as
$\bigwedge_{i=1}^{r} D(X_i, \bar{X}_i) \wedge \bigwedge_{i=1}^{s} D(Y_i, \bar{Y}_i) \wedge \bigwedge_{i=1}^{n} C(L^*_{i,1}, L^*_{i,2}, L^*_{i,3}) \rightarrow ans(X_1, \ldots, X_r)$,

where $L^*_{i,j} = \begin{cases} X_\beta & \text{if } l_{i,j} = x_\alpha \text{ and } x_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ \bar{X}_\beta & \text{if } l_{i,j} = \neg x_\alpha \text{ and } x_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ Y_\beta & \text{if } l_{i,j} = y_\alpha \text{ and } y_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ \bar{Y}_\beta & \text{if } l_{i,j} = \neg y_\alpha \text{ and } y_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \end{cases}$

The intuition behind this reduction is as follows:

For $\hat{G} = Cl_{\mathcal{R}}(G)$, $q(\hat{G}) = \{0,1\}^r$, as can be easily checked. Therefore r is only redundant w.r.t. q if also $q(G) = \{0,1\}^r$. Now this is exactly the case if F is satisfiable:

If F is satisfiable, then there exists for every truth assignment on $\vec{x}_1$ a truth assignment on $\vec{y}_1$ s.t. $\bigwedge_{i=1}^{n} C_i$ evaluates to $true$. Given an arbitrary homomorphism that maps the first group of conjuncts into G, we can define from this mapping a truth assignment on $\vec{x}_1$, from which we know that it can be extended to a truth assignment on $\vec{y}_1$ s.t. no clause C_i evaluates to $false$ (i.e., in now clause all three literals get assigned false). We can now translate this truth assignment on $\vec{y}_1$ to a homomorphism for the second group of conjuncts. By the structure of the third group of conjuncts, we can also be sure that none of the tuples $C(.,.,.)$ in the body of the query is mapped to $C(0,0,0)$, and therefore q returns the same answer also over G.

On the other hand, if we now that $q(G) = q(\hat{G})$, given an arbitrary truth assignment on $\vec{x}_1$, it translates into one answer to $q(G)$. Therefore we know that there exists a homomorphism from $body(q)$ to G. As G does not contain $C(0,0,0)$, we can translate this homomorphism into a truth assignment on F s.t. every C_i evaluates to $true$. □

D Full proofs for Section 6

First note that the intuitive explanation from Section 6 for why the complexity of MINI-RDF^{card} for cases that are already NP-hard for the problems considered in Section 3, can now be verified using the algorithms presented in the membership proofs for Section 3: Obviously, in these cases step (1) can be always replaced by "Guess a subgraph with at most k triples" without affecting the overall complexity. On the other hand, the problem also cannot be easier, as this would give an immediate contradiction to the NP-hardness of the settings considered in Section 3.

Theorem 6.1. *The problem* MINI-RDF$^{card}(G, \mathcal{R}, \mathcal{C}, k)$ *is* NP*-complete if $\mathcal{C} = \emptyset$ and $\mathcal{R}$ is either considered as fixed or a set of b-bounded rules (for fixed b).*

Proof. We show the membership for $\mathcal{R}$ being a set of b-bounded datalog rules and the hardness for a fixed set of datalog rules.
The following algorithm proves the NP-*membership* by deciding MINI-RDF^{card}

(1) Guess $G' \subset G$ with $|G'| \leq k$
(2) Compute $Cl_{\mathcal{R}}(G')$
(3) Return "no" if for some tuple $t \in G$, $t \notin Cl_{\mathcal{R}}(G')$ holds, otherwise return "yes".

Step (2) requires only polynomial time ($\mathcal{R}$ is b-bounded; for a more detailed justification that this allows for a polynomial algorithm for computing the closure see e.g. the proof of Theorem 4.2). Note that in step (1), $|G'| \leq k$ can be replaced by $< k$ or $= k$ without affecting the complexity.

In the main body of the text, we only gave an informal description of the idea of the reduction. It was left open to present the reduction formally and to show its correctness. Recall that the reduction is from *Vertex Cover*.

Consider a fixed set $\mathcal{R}$ of rules:
$\mathcal{R} = \{\ \{V\ neighbour\ E\,.\,V\ v\ V\} \Rightarrow \{E\ e\ E\};$
$\{E\ e\ E\,.\,X\ in\ X\,.\,X\ succ\ E\} \Rightarrow \{E\ in\ E\};$
$\{E\ last\ E\,.\,E\ in\ E\,.\,V\ backupv\ V\} \Rightarrow \{V\ v\ V\}\ \}.$
Now let an arbitrary instance of Vertex Cover be given by a graph $G = (V, E)$ and an integer k'. W.l.o.g. we assume G to contain no selfloops, i.e. edges (v_i, v_i). We define an instance of MINI-RDF^{card} $(G^{rdf}, \mathcal{R}, \mathcal{C}, k)$ as follows: $\mathcal{R}$ is defined as the set of rules given above, $\mathcal{C} = \emptyset$ and $G^{rdf} = G^v \cup G^{nghb} \cup G^{ord}$, where the partitions contain the following triples:
$G^v = \{v_i\ v\ v_i\,.\,v_i\ backupv\ v_i \mid v_i \in V\}$, where we introduce a new URI v_i for every $v_i \in V$ and, by slight abuse of notation, use v_i to denote both, the vertex in the input graph and the URI in the RDF graph.
$G^{nghb} = \{v_i\ neighbour\ e_\ell \mid$ for every $v_i \in V$ and $e_\ell \in E$ with $e_\ell = (v_i, v_j)$ or $e_\ell = (v_j, v_i)\}$, where we again use e_ℓ to denote both, edges from the input graph and a new URI for every edge.
For defining G^{ord}, assume an arbitrary order on the edges in E (i.e. let $E = (e_1, \ldots, e_m)$ if $|E| = m$), and set $G^{ord} = \{e_i\ succ\ e_{i+1} \mid i \in \{1, \ldots, m-1\}\} \cup \{e_m\ last\ e_m\,.\,e_0\ in\ e_0\,.\,e_0\ succ\ e_1\}$ where e_0 is some fresh URI, representing some "dummy" edge. Finally we set $k = 3 * m + 2 + k' + |V|$.

The reduction is obviously feasible in LOGSPACE. Before showing that it is indeed correct, we first sketch its intuition. Note that the only triples from G^{rdf} that can be derived by $\mathcal{R}$ are such of the from $v_i\ v\ v_i$. Therefore the idea is that those triples with v on predicate position remaining in some valid subgraph G' encode a valid vertex cover. To ensure this, the first rule in $\mathcal{R}$ adds a triple $e_j\ e\ e_j$ for every edge covered by G'. The second rule adds $e_j\ in\ e_j$ to G' if all predecessors of e_j according to the assumed arbitrary order are covered by G'. Hence if $e_m\ in\ e_m$ can be derived for the last edge e_m in this order, then

all edges are indeed covered. If this is the case, the last rule allows to re-insert again the triples $v_i\ v\ v_i$ for all vertices not being part of the vertex cover, hence the complete graph G^{rdf}.

Finally we show that the reduction is indeed correct, i.e. that there exists a vertex cover of size k' iff there exists a subgraph G' containing $3*m+2+k'+|V|$ triples, s.t. $G^{rdf} \subseteq Cl_{\mathcal{R}}(G')$ and $|G'| = k$. Before we show the two directions separately, we first comment on the size of G^{rdf}, which is $2*m+|V|+|V|+2+m$, where $m = |E|$. This number is derived as follows: Because every edge connects two different nodes (we assume no selfloops in G), G^{rdf} contains $2*m$ triples of the form $\{v_i\ neighbour\ e_k\}$. It further contains $|V|$ triples of the form $\{v_i\ backupv\ v_i\}$, $|V|$ triples of the form $\{v_i\ v\ v_i\}$, the two triples $\{e_m\ last\ e_m\,.\,e_0\ in\ e_0\}$, and m triples $\{e_i\ succ\ e_{i+1}\}$ (for $i \in \{0,\ldots,m-1\}$).

"If" direction) Assume that such a G' exists. We have to show that then there exists a vertex cover of size k'. First note that the only triples that can be removed from G^{rdf} such that $G^{rdf} \subseteq Cl_{\mathcal{R}}(G')$ holds are triples of the form $v_i\ v\ v_i$. Denoting the number of such triples that remain in G' with r, the size of G' is obviously $2*m+|V|+2+m+r$. (I.e. G' differs by $|V|-r$ triples of the form $v_i\ v\ v_i$ from G^{rdf}.)

Now we define a vertex cover of G to contain exactly those nodes whose corresponding triples $v_i\ v\ v_i$ are in G', i.e. $VC = \{v_i \mid v_i\ v\ v_i \in G'\}$. As by assumption $|G'| = k = 3*m+2+k'+|V|$, it follows that $k' = r$, hence VC has the required size. It remains to show that VC is indeed a valid vertex cover: Because G' satisfies $G^{rdf} \subseteq Cl_{\mathcal{R}}(G')$, we know that the third rule is applicable, hence that $e_m\ in\ e_m \in Cl_{\mathcal{R}}(G')$ (remember the assumed arbitrary ordering on E). By induction along this order, it is easy to check that $e_j\ in\ e_j \in Cl_{\mathcal{R}}(G')$ only holds if VC covers all edges $e_\ell \leq e_j$ (again $\leq$ w.r.t. the the ordering on E). To see this, just note that $e_\ell\ in\ e_\ell \in Cl_{\mathcal{R}}(G')$ iff $\{e_{\ell-1}\ in\ e_{\ell-1}\,.\,e_\ell\ e\ e_\ell\} \subset Cl_{\mathcal{R}}(G')$, and that for every e_ℓ the triple $e_\ell\ e\ e_\ell$ is in $Cl_{\mathcal{R}}(G')$ only if for a vertex v_i adjacent to e_ℓ, $v_i\ v\ v_i$ is in G'. Therefore from $\{e_m\ in\ e_m\} \subset Cl_{\mathcal{R}}(G')$ we derive that VC indeed covers all $e_j \in E$.

"Only-if" direction) Assume that there exists a Vertex Cover $VC \subseteq V$ with $|VC| \leq k'$. We have to show that then there exists a subgraph G' of size k s.t. $G^{rdf} \subseteq Cl_{\mathcal{R}}(G')$ holds (as $\mathcal{C} = \emptyset$ is trivially satisfied). We claim that $G' = G^{rdf} \setminus \{v_i\ v\ v_i \mid v_i \in V \setminus VC\}$ is the desired subgraph: Since VC is a valid vertex cover, for every edge at least one of its endpoints lies in VC, say v_i. Hence $v_i\ v\ v_i \in G'$. Therefore $\{e_\ell\ e\ e_\ell \mid e_\ell \in E\} \subset Cl_{\mathcal{R}}(G')$, by the first rule. Because of this, also $\{e_\ell\ in\ e_\ell \mid e_\ell \in E\} \subset Cl_{\mathcal{R}}(G')$, which finally allows us to derive $\{v_i\ v\ v_i \mid v_i \in V\} \subset Cl_{\mathcal{R}}(G')$, which proves the case. The size $|G'| = 3m+2+|V|+k'$ follows trivially from the definition of G' and the size of G^{rdf}. □ □

Theorem 6.2. *Let G be a RDF graph and $\mathcal{C}$ a set of tgds. Deciding whether there exists some $\emptyset \neq G' \subset G$ s.t. G' satisfies $\mathcal{C}$ is Σ_3^P-complete.*

Proof. In the main body of the text, it was already mentioned that this theorem is shown to be correct by a modification of the proof of Lemma 3.4, where all information implicitly expressed in the set of rules has to be expressed explicitly as tgds. Therefore, we arrive at the following reduction:

Let an arbitrary instance of QSAT$_3$ be given by $F = \exists\vec{x}_1\forall\vec{y}_1\exists\vec{x}_2 \bigwedge_{i=1}^{n} C_i$, with $C_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$, where $l_{i,j}$ is a literal. Let $|\vec{x}_1| = r$, $|\vec{y}_1| = s$ and $|\vec{x}_2| = t$ and let V denote the set $V = \vec{x}_1 \cup \vec{x}_2 \cup \vec{y}_1$ where we write $V = \{v_i \mid i \in [r+s+t]\}$ and $v_i \in \begin{cases} \vec{x}_1 & \text{if } i \leq r \\ \vec{y}_1 & \text{if } i > r \text{ and } i \leq r+s \\ \vec{x}_2 & \text{if } i > r+s \end{cases}$

By slight abouse of notation, we use v_i also to denote URIs in the RDF graph. Then we define $G = G_{01} \cup G_{000} \cup G_{q1} \cup G_{q4} \cup G_{q2} \cup G_{opt}$ and $\mathcal{C}$ as follows:

$G_{01} = \{0\ h_1\ a_{001}\,.\,0\ h_2\ a_{001}\,.\,1\ h_3\ a_{001}\ .\ 0\ h_1\ a_{010}\,.\,1\ h_2\ a_{010}\,.\,0\ h_3\ a_{010}\ .$
$1\ h_1\ a_{100}\,.\,0\ h_2\ a_{100}\,.\,0\ h_3\ a_{100}\ .\ 0\ h_1\ a_{011}\,.\,1\ h_2\ a_{011}\,.\,1\ h_3\ a_{011}\ .$

$1\ h_1\ a_{101}\,.\,0\ h_2\ a_{101}\,.\,1\ h_3\ a_{101}\ .\ 1\ h_1\ a_{110}\,.\,1\ h_2\ a_{110}\,.\,0\ h_3\ a_{110}\ .$
$1\ h_1\ a_{111}\,.\,1\ h_2\ a_{111}\,.\,1\ h_3\ a_{111}\}$
$G_{000} = \{0\ h_1\ a_{000}\,.\,0\ h_2\ a_{000}\,.\,0\ h_3\ a_{000}\}$
$G_{q1} = \{v_i\ q_1\ a_{01}\,.\,v_i\ q_1\ a_{10} \mid i \leq r\}$
$G_{q4} = \{v_i\ q_4\ a_{01}\,.\,v_i\ q_4\ a_{10} \mid r < i \leq r+t\}$
$G_{q2} = \{0\ q_2\ a_{01}\,.\,1\ q_2\ a_{10}\,.\,1\ q_3\ a_{01}\,.\,0\ q_3\ a_{10}\}$
$G_{opt} = \{v_i\ opt\ v_i \mid i \leq r\}$

Note that G contains all the triples from the graph in the proof of Lemma 3.4, but in addition the set of triples $\{v_i\ opt\ v_i \mid i \leq r\}$. By adding tgds enforcing these triples to remain in every valid subgraph, these triples are used to express that exactly one of the triples $\{v_i\ q_1\ a_{01}\,.\,v_i\ q_1\ a_{10}\}$ for every $v_i \in \vec{x}_1$ has to remain in every valid subgraph.

The set of constraints is defined as

$\mathcal{C} = \{\{0\ P\ a_{000}\} \Rightarrow \{\tau\} \mid \tau \in G\} \cup$
$\{\{R\ q_1\ a_{10}\,.\,R\ q_1\ a_{01}\} \Rightarrow G_{000}\} \cup$
$\{\{A\ B\ C\} \Rightarrow \{\tau\} \mid \tau \in G_{01} \cup G_{q4} \cup G_{q2} \cup G_{opt}\} \cup$
$\{\{V\ opt\ V\} \Rightarrow \{V\ q_1\ A\}\} \cup$
$\{\forall \vec{A_1}, \vec{X}, \vec{\bar{X}}, \vec{Y}, \vec{\bar{Y}}\ \phi \Rightarrow \exists \vec{A_2}, \vec{Z}, \vec{\bar{Z}}, \vec{R}\ \psi\}$ where

$\vec{A_1} = A_1, \ldots, A_{r+s}$, $\vec{A_2} = A_{r+s+1}, \ldots, A_{r+s+t}$,
$\vec{X} = X_1, \ldots, X_r$, $\vec{\bar{X}} = \bar{X}_1, \ldots, \bar{X}_r$,
$\vec{Y} = Y_1, \ldots, Y_s$, $\vec{\bar{Y}} = \bar{Y}_1, \ldots, \bar{Y}_s$,
$\vec{Z} = Z_1, \ldots, Z_t$, $\vec{\bar{Z}} = \bar{Z}_1, \ldots, \bar{Z}_t$,
$\vec{R} = R_1, \ldots, R_n$ and
$\phi = \{v_i\ q_1\ A_i\,.\,X_i\ q_2\ A_i\,.\,\bar{X}_i\ q_3\ A_i \mid i \in [r]\} \cup$
$\{v_i\ q_4\ A_i\,.\,Y_j\ q_2\ A_i\,.\,\bar{Y}_j\ q_3\ A_i \mid i \in \{r+1, \ldots, r_s\}$ and $j = i - r\}$ and
$\psi = \{v_i\ q_4\ A_i\,.\,Z_j\ q_2\ A_i\,.\,\bar{Z}_j\ q_3\ A_i \mid i \in \{r+s+1, \ldots, r+s+t$ and $j = i - r - s\}\} \cup$
$\{L^*_{i,1}\ h_1\ R_i\,.\,L^*_{i,2}\ h_2\ R_i\,.\,L^*_{i,3}\ h_3\ R_i \mid i \in [n]\}$

where $L^*_{i,j} = \begin{cases} X_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ \bar{X}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_1 \text{ and } \beta = \alpha \\ Y_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ \bar{Y}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{y}_1 \text{ and } \beta = \alpha - r \\ Z_\beta & \text{if } l_{i,j} = v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \\ \bar{Z}_\beta & \text{if } l_{i,j} = \neg v_\alpha \text{ and } v_\alpha \in \vec{x}_2 \text{ and } \beta = \alpha - r - s \end{cases}$

Obviously, this reduction is feasible in LOGSPACE.

First note, that G satisfies $\mathcal{C}$. It remains to show that F is satisfiable iff there exists some $G' \subset G$ s.t. $G' \neq \emptyset$ and G' satisfies $\mathcal{C}$. We show both directions separately.

"If" - direction) Assume that there exists such a G'. We have to show that then F is satisfied. First we note that every nonempty proper subset G' of G that satisfies $\mathcal{C}$ must not contain G_{000} (because of the first tgd), but has to contain $G_{01} \cup G_{q4} \cup G_{q2} \cup G_{opt}$ (because of the third tgd), and for every $i \leq r$ exactly one of the two triples $\{v_i\ q_1\ a_{01}\,.\,v_i\ q_1\ a_{10}\}$ (because of the second, first, and forth tgd). Let G' be an arbitrary such subgraph, and define a truth assignment T_1 on the variables $\vec{x}_1$ as follows: $T_1(v_i) = \mathit{true}$ if $\{v_i\ q_1\ a_{10}\} \subseteq G'$, and $T_1(v_i) = \mathit{false}$ if $\{v_i\ q_1\ a_{01}\} \subseteq G'$ ($i \leq r$). Because of the reasons stated before, this gives a well defined truth assignment. We claim that T_1 is a truth assignment on $\vec{x}_1$, such that for every truth assignment on $\vec{y}_1$, there exists a truth assignment on $\vec{x}_2$ such that $\bar{F} = \bigwedge_{i=1}^{n} C_i$ evaluates

to $true$. To see this, let T_2 be an arbitrary truth assignment on $\vec{y}_1$. From this, we define a homomorphism $\mu\colon \phi \to G'$ as follows: $\mu(X_i) = 1$ if $T_1(v_i) = true$, and $\mu(X_i) = 0$ if $T_1(v_i) = false$. Similarily, $\mu(Y_{i-r}) = 1$ if $T_2(v_i) = true$, and $\mu(Y_{i-1}) = 0$ if $T_2(v_i) = false$. Moreover, $\mu(\bar{X}_i) = 1 - \mu(X_i)$ and $\mu(\bar{Y}_{i-r}) = 1 - \mu(Y_{i-r})$. Finally $\mu(A_i) = a_{10}$ if $i \leq r$ and $T_1(v_i) = true$, $\mu(A_i) = a_{01}$ if $i \leq r$ and $T_1(v_i) = false$, and similarily $\mu(A_i) = a_{10}$ if $r < i \leq r + s$ and $T_2(v_i) = true$, $\mu(A_i) = a_{01}$ if $r < i \leq r + s$ and $T_2(y_i) = false$. It is easy to check that this gives indeed a valid homomorphism from ϕ to G'. Now, as we assume that G' satisfies $\mathcal{C}$, there exists at least one extension μ' of μ, that maps also ψ to G' (let μ' be an arbitrary such extension). From this homomorphism, we define the following truth assignment on $\vec{x}_2$: For every $v_i \in \vec{x}_2$, we define $T_3(v_i) = true$ if $\mu'(Z_{i-r-s}) = 1$, and $T_3(v_i) = false$ if $\mu'(Z_{i-r-s}) = 0$. It goes without saying that this gives a well defined truth assignment. It remains to show that $\bar{F}$ indeed evaluates to $true$ under $T = T_1 \cup T_2 \cup T_3$. This is the case if for every C_i in F, at least one literal evaluates to $true$. To see that this holds, note that for every $i \in [n]$, for every value of $\mu'(R_i)$ it cannot be that $\mu'(L^*_{i,1}) = \mu'(L^*_{i,2}) = \mu'(L^*_{i,3}) = 0$. (As for no value a_{xyz} there exist three triples in G', one with h_1 on predicate position, one with h_2 and one with h_3 such that all have a_{xyz} on the object position and 0 on subject position.) But by definition $L^*_{i,j} = X_\alpha$ (resp. Y_α, Z_α) if $l_{i,j}$ is a positive literal. Hence if $L^*_{i,j} = 1$, $T(l_{i,j}) = true$. Also, $L^*_{i,j} = \bar{X}_\alpha$ (resp. $\bar{Y}_\alpha$, $\bar{Z}_\alpha$) if $l_{i,j}$ is a negative literal. Hence again $T(l_{i,j}) = true$, because if $l_{i,j} = \neg v_\alpha$, then $T(v_\alpha) = false$. Thus F is satisfied.

"Only if"-direction) Assume that F is satisfied. We have to show that then there exists a valid sugraph G'. F is satisfied if there exists a truth assignment T_1 on $\vec{x}_1$ such that for every truth assignment T_2 on $\vec{y}_1$ there exists a truth assignment T_3 on $\vec{x}_2$ such that F evaluates to $true$. Let T_1 be an arbitrary such truth assignment. Then we define a subset $G' \subset G$ as follows: $G' = G_{01} \cup G_{q4} \cup G_{q2} \cup G_{opt} \cup G_2$, where G_2 contains for every $i \leq r$ the triple $\{v_i\ q_1\ a_{10}\}$ if $T_1(v_i) = true$, and $\{v_i\ q_1\ a_{01}\}$ if $T_1(v_i) = false$. Obviously, G' is a proper subset of G. It further satisfies the first four tgds from $\mathcal{C}$: It contains every triple from G_1; for every $i \leq r$ (exactly for those i, there exists a triple $\{v_i\ opt\ v_i\}$ in G_1) it contains one of the two triples $\{v_i\ q_1\ a_{10}\,.\,v_i\ q_1\ a_{01}\}$; but not both, and also not G_{000}, hence the antecedent of the first two tgd is not satisfied. To show that also the last tgd is satisfied, let μ be an arbitrary homomorphism from ϕ to G'. From this, we define a truth assignment T_2 on $\vec{y}_1$ as follows: Let $T_2(v_i) = true$ if $i > r$ and $\mu(Y_{i-r}) = 1$, and $T_2(v_i) = false$ if $i > r$ and $\mu(Y_{i-r}) = 0$. This obviously gives a well defined truth assignment, as μ maps every Y_{i-r} to exactly one value from $\{0, 1\}$ (all triples in G' with q_2 or q_3 on the second position have either 0 or 1 on the first position). Note that if we define an according truth assignment for $\vec{x}_1$, this gives exactly T_1. From the assumption we know that for T_1 and every truth assignment T_2 on $\vec{y}_1$, there exists a truth assignment T_3 on $\vec{x}_2$ s.t. F evaluates to $true$. Let T_3 be such an assignment. Now we define an extension μ' of μ to $\vec{A}_2$, $\vec{Z}$, $\vec{\bar{Z}}$ and $\vec{R}$ as follows: For $i \in \{r + s + 1, \ldots, r + s + t\}$, let $\mu'(Z_{i-r-s}) = 1$ and $A_i = a_{10}$ if, $T_3(v_i) = true$, and $\mu'(Z_{i-r-s}) = 0$ and $A_i = a_{10}$ if $T_3(v_i) = false$. Moreover, $\mu'(\bar{Z}_{i-r-s}) = 1 - \mu'(Z_{i-r-s})$. And finally, for every clause C_i ($i \in [n]$) in F, we define $\mu'(R_i) = a_{xyz}$, where x (y,z. resp.) is 0 if $T(l_{i,1})$ (resp. $l_{i,2}$, $l_{i,3}$) is $false$ and 1 if it is $true$. Thereby $T = T_1 \cup T_2 \cup T_3$. It remains to show that $\mu'(\psi) \subseteq G'$. As the triples $\{v_i\ q_4\ A_i\ .\ Z_{i-r-s}\ q_2\ A_i\ .\ \bar{Z}_{i-r-s}\ q_3\ A_i\}$ are obviously mapped to G', only the triples $\{L^*_{i,1}\ h_1\ R_i\,.\,L^*_{i,2}\ h_2\ R_i\,.\,L^*_{i,3}\ h_3\ R_i \mid i \in [n]\}$ remain. For them being also mapped into G', it is important that the values for $L^*_{i,j}$ and R_i "fit". To see that this is the case, just note that, by definition, $\mu'(L^*_{i,j}) = 1$ iff $T(l_{i,j}) = true$, as can be easily checked. But then, $\mu'(L^*_{i,j})$ contains exactly the value encoded in the name of a_{xyz} in the first (resp. second or third) position. Now it is easy to see, that for every pair (h_1, a_{xyz}), G' contains a triple $\{x\ h_1\ a_{xyz}\}$, and the same holds for h_2 and h_3, which proves the case. As μ' maps ψ to G', this proves that also the last tgd is satisfied by G', and we are done. □ □

Theorem 6.3. *Let G be a RDF graph and $\mathcal{C}$ a set of b-bounded tgds. Deciding whether there exists some*

$\emptyset \neq G' \subset G$ *s.t.* $G' \models \mathcal{C}$ *is* NP*-complete.*

Proof. We only prove the NP-hardness. The proof is by reduction from SAT: Let an arbitrary instance of SAT be given by the formula $F = \bigwedge_{i=1}^{n} C_i$ with $C_i = (l_{i,1} \vee \cdots \vee l_{i,k_i})$ (where $l_{i,j}$ are literals). We assume that every variable in F occurs negated and unnegated. Introducing two new URIs x_i and $\bar{x}_i$ for every variable x_i in F, and one new URI c_i for every clause in F, we define G and $\mathcal{C}$ as follows:

$$
\begin{aligned}
G = {} & \{l^*_{i,j}\ \mathit{in}\ c_i \mid i \in [n], j \in [k_i]\} \cup \\
& \{l^*_{i,j}\ \mathit{active}\ c_i \mid i \in [n], j \in [k_i]\} \cup \\
& \{x_j\ \mathit{clash}\ \bar{x}_j \mid x_j \text{ in } F\} \cup \\
& \{c_i\ \mathit{clause}\ c_i \mid i \in [n]\} \text{ and} \\
\mathcal{C} = {} & \big\{\ \{X\ \mathit{active}\ I\,.\,X\ \mathit{in}\ J\} \Rightarrow \{X\ \mathit{active}\ J\}; \\
& \quad \{X\ \mathit{clash}\ X'\,.\,X\ \mathit{active}\ I\,.\,X'\ \mathit{active}\ I'\,.\,Y\ \mathit{in}\ J\} \Rightarrow \{Y\ \mathit{active}\ J\}; \\
& \quad \{I\ \mathit{clause}\ I\} \Rightarrow \{X\ \mathit{active}\ I\}\big\} \cup \\
& \quad \{\{A\ B\ C\} \Rightarrow \{c_i\ \mathit{clause}\ c_i\} \mid i \in [n]\} \cup \\
& \quad \{\{A\ B\ C\} \Rightarrow \{l^*_{i,j}\ \mathit{in}\ c_i\} \mid i \in [n], j \in [k_i]\} \cup \\
& \quad \{\{A\ B\ C\} \Rightarrow \{x_j\ \mathit{clash}\ \bar{x}_j\} \mid x_j \text{ in } F\}
\end{aligned}
$$

where $l^*_{i,j} = x_\alpha$ if $l_{i,j} = x_\alpha$ and $l^*_{i,j} = \bar{x}_\alpha$ if $l_{i,j} = \neg x_\alpha$.
This reduction is obviously feasible in LOGSPACE. It remains to show that F is satisfiable iff there exists a subgraph G' of G s.t. G' satisfies $\mathcal{C}$. We show both directions separately.

"If"-direction) Assume that there exists such a G'. We have to show that then F is satisfiable. We first note that the only triples from G not in G' are triples with active on the predicate position. This is because $G' \neq \emptyset$, and as long as G' contains some triple, because G' satisfies $\mathcal{C}$, it has to contain all triples from G with clause, in or clash on their predicate position. Now we define a truth assignment T on the variables in F as follows: If for a variable x_j in F, $\{x_j\ \mathit{active}\ c_i\}$ is in G' (for some clause C_i), then $T(x_j) = \mathit{true}$, and if $\{\bar{x}_j\ \mathit{active}\ c_i\}$ is in G', then $T(x_j) = \mathit{false}$. If neither the one nor the other kind of triple is contained in G', then, w.l.o.g., set $T(x_j) = \mathit{false}$. Under the assumption that T is indeed well defined, it is easy to see that F evaluates to true under T: Because of the third tgd, for every clause C_i, a triple $\{X\ \mathit{active}\ c_i\}$ must be in G'. Now if X is some x_j, then x_j appears unnegated in C_i, hence C_i evalutes to true under T (as $T(x_j) = \mathit{true}$ by the above definition). If X is some $\bar{x}_j$, then x_j appears negated in C_i, but again C_i evaluates to true under T, as $T(x_j) = \mathit{false}$ by the above definition of T. Hence it remains to show that T is indeed well defined: It follows directly from the definition that T assigns to each variable at least one truth value. To see that this assignment is indeed unique for every variable, assume to the contrary that this is not the case. This would require for some variable x_j, that there exists a triple $\{x_j\ \mathit{active}\ c_i\}$ in G' and another one $\{\bar{x}_j\ \mathit{active}\ c_\ell\}$. But G' also contains a triple $\{x_j\ \mathit{clash}\ \bar{x}_j\}$, and therefore, to satisfy the second tgd, G' had to contain all triples of the form $\{l^*_{r,s}\ \mathit{active}\ c_r\}$ (for all $r \in [n], s \in [k_i]$). However, this is equal to $G' = G$, which is a contradiction to the assumption that G' is a real subgraph $G' \subset G$. This shows that T is indeed well defined, which proves the case.

"Only if"-direction) Assume that F is satisfiable. It remains to thow that then there exists an appropriate G'. We define G' as follows: For every variable x_α in F, let

$$
G_{x_\alpha} = \begin{cases} \{x_\alpha\ \mathit{active}\ c_\ell \mid \text{all clauses } c_\ell \text{ containing } x_\alpha\} & \text{if } T(x_\alpha) = \mathit{false} \\ \{\bar{x}_\alpha\ \mathit{active}\ c_\ell \mid \text{all clauses } c_\ell \text{ containing } \bar{x}_\alpha\} & \text{if } T(x_\alpha) = \mathit{true} \end{cases}
$$

Then we define $G' = G \setminus \{G_{x_\alpha} \mid \text{for all } x_\alpha \in F\}$. Obviously, all tgds with the antecedent $\{A\ B\ C\}$ are satisfied by G'. Moreover, as by assumption every variable occurs at least once negated and once unnegated in F, $G_{x_\alpha} \neq \emptyset$, hence G' is a real subgraph of G. It remains to show that each of the first three tgds is satsified by G'.

$\{X\ active\ I\,.\,X\ in\ J\} \Rightarrow \{X\ active\ J\}$: By the definition of G', a triple $\{X\ active\ c_i\}$, where X is either some x_j or $\bar{x}_j$, is only removed from G if $T(x_j) = false$ (resp. $true$). But then, all triples with the URI x_j (resp. $\bar{x}_j$) in subject position are removed from G. Hence, either both, antecendent and consequent can be mapped to G', or none of them.
$\{X\ clash\ X'\,.\,X\ active\ I\,.\,X'\ active\ I'\,.\,Y\ in\ J\} \Rightarrow \{Y\ active\ J\}$: As G' is a real subset of G, this tgd is violated as soon as for some triple $\{x_j\ clash\ \bar{x}_j\}$, for both, x_j and $\bar{x}_j$ there exists a triple $\{x_j\ active\ c_{i_1}\}$, resp. $\{\bar{x}_j\ active\ c_{i_2}\}$ in G'. However, by definition, $\{x_j\ active\ c_{i_1}\}$ is in G' only if $T(x_j) = true$, while $\{\bar{x}_j\ active\ c_{i_2}\}$ is in G' only if $T(x_j) = false$. As T is a well defined truth assignment, this is a contradiction, hence the tgd is satisfied.
$\{I\ clause\ I\} \Rightarrow \{X\ active\ I\}$: As T satisfied F, in every clause C_i of F at least one literal evaluates to $true$ under F. If this literal is positive (e.g. x_j), then some triple $\{x_j\ active\ c_i\}$ remains in G' by definition. If it is a negative literal (e.g. $\neg x_j$), then $T(x_j) = false$ and $\{\bar{x}_j\ active\ c_i\}$ remains in G' by definition. However, in both cases the tgd this satisfied. □ □

Theorem 6.4. *The problems* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ *are Σ_2^P-complete if $\mathcal{C}$ is a set of full tgds. Σ_2^P-completeness even holds for fixed $\mathcal{R}$.*

Likewise, let G be an RDF graph and $\mathcal{C}$ a set of full tgds. Deciding whether there exists some $\emptyset \neq G' \subset G$ s.t. G' satisfies $\mathcal{C}$ is Σ_2^P-complete.

Proof. We only prove the Σ_2^P-completeness for the MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ and MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ problems. The other Σ_2^P-completeness result (i.e., checking if some $\emptyset \neq G' \subset G$ exists, s.t. G' satisfies $\mathcal{C}$) is shown analogously.

The Σ_2^P-membership is established by the same algorithm as in Lemma A.1. Note that there the check if $G' \not\models c$ holds for every $c \in \mathcal{C}$ required a Π_2^P-oracle. Now that we are restricting ourselves to full tgds, this check is feasible by a coNP-oracle, from which the Σ_2^P-membership follows immediately.

The Σ_2^P-hardness is shown via reduction from QSAT$_2$. We only present the reduction in terms of a schema S containing also ternary relations, namely $S = \{V/3, D/2, C/3, C'/3, R/0, R'/0\}$. The translation into RDF is analogous to previous translations (e.g., Lemma 3.4) and is therefore omitted.

Let an arbitrary instance of QSAT$_2$ be given by $F = \exists\vec{x}_1 \forall\vec{y}_1 \bigvee_{i=1}^{n} D_i$, where $D_i = (l_{i,1} \wedge l_{i,2} \wedge l_{i,3})$ (obviously, the restriction to 3-DNF goes w.l.o.g.) and where $\vec{x}_1 = \{x_1, \ldots, x_k\}$ and $\vec{y}_1 = \{y_1, \ldots, y_\ell\}$. From this we define G, $\mathcal{R}$, and $\mathcal{C}$ as follows:

$$
\begin{aligned}
G = {} & \{V(x_i, 0, 1), V(x_i, 1, 0) \mid x_i \in \vec{x}_1\} \cup \{D(0,1), D(1,0)\} \cup \\
& \{C(0,0,0), C(0,0,1), C(0,1,0), C(0,1,1), \cup \\
& \{C(1,0,0), C(1,0,1), C(1,1,0), C(1,1,1)\} \cup \\
& \{C'(1,1,1), R(), R'()\}.
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{R} = {} & \{\{V(X,Y,Z)\} \Rightarrow \{V(X,Z,Y)\}, \\
& \{C'(X,Y,Z)\} \Rightarrow \{C(X,Z,Y)\}, \\
& \{R'()\} \Rightarrow \{R()\}\}.
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{C} = {} & \{\{R(), V(X,Y,Z)\} \Rightarrow \{V(X,Z,Y)\}, \\
& \{R(), C'(X,Y,Z)\} \Rightarrow \{C(X,Y,Z)\}, \\
& \{C(X,Y,Z), C'(X,Y,Z)\} \Rightarrow \{R()\}, \\
& \{V(X,Y,Z), V(X,Z,Y)\} \Rightarrow \{R()\}, \\
& \{C(L^*_{i,1}, L^*_{i,2}, L^*_{i,3}) \mid 1 \leq i \leq n\} \cup \{V(x_1, X_i, \bar{X}_i) \mid 1 \leq i \leq k\} \cup \\
& \quad \{D(Y_i, \bar{Y}_i) \mid 1 \leq i \leq \ell\} \Rightarrow \{R()\}, \text{ s.t.}
\end{aligned}
$$

$$L^*_{i,j} = \begin{cases} X_\alpha & \text{if } l_{i,j} = x_\alpha \\ \bar{X}_\alpha & \text{if } l_{i,j} = \neg x_\alpha \\ Y_\beta & \text{if } l_{i,j} = y_\beta \\ \bar{Y}_\beta & \text{if } l_{i,j} = \neg y_\beta \end{cases}$$

The idea of this reduction is as follows: Let G' be a proper subgraph of G, s.t. $Cl_{\mathcal{R}}(G') \supseteq G$. Then the rules in $\mathcal{R}$ make sure that $G \setminus G'$ may contain the following atoms: By the first rule, for each x_i, at most least one of $V(x_i, 0, 1), V(x_i, 1, 0)$ may be in $G \setminus G'$. Moreover, by the last two rules, $G \setminus G'$ may possibly contain $C(1, 1, 1)$ and $R()$. The first four constraints in $\mathcal{C}$ make sure that $G \setminus G'$ indeed does contain one of $V(x_i, 0, 1), V(x_i, 1, 0)$ for each x_i as well as the atoms $C(1, 1, 1)$ and $R()$: By the first two constraints, if G' contains $R()$, then $G = G'$ must hold. By the third constraint, if G' contains $C(1, 1, 1)$, then G' also contains $R()$ and, therefore, again $G = G'$ must hold. The fourth constraint makes sure that, for each x_i, if G' contains both $V(x_i, 0, 1), V(x_i, 1, 0)$, then $G' = G$.

The idea of the last constraint is as follows: Suppose that G' is a proper subgraph of G. By the above considerations, $R()$ is not contained in G'. Hence, the only way that G' satisfies the last constraint is that there exists no homomorphism from the body of this constraint to G'. As in the proof of Lemma 3.4, every homomorphism from the body of this constraint to G' defines a truth assignment on the propositional variables in F. In other words, such a proper subgraph G' exists, if the following condition holds:

For each i, there exists exactly one of $V(x_i, 0, 1)$ and $V(x_i, 1, 0)$ in G' (in terms of the formula F, this means that there exists a truth assignment I on the variables in $\vec{x}_1$), s.t. for any homomorphism from $\{D(Y_1, \bar{Y}_1), \ldots, D(Y_\ell, \bar{Y}_\ell)\}$ (i.e., for any extension J of the assignment I to the variables in $\vec{y}_1$), there exists at least one atom $C(L^*_{i,1}, L^*_{i,2}, L^*_{i,3})$ which is not mapped to an atom in G' by this homomorphism, i.e.: this atom is mapped to $C(1, 1, 1)$ (in terms of the formula F, this means that at least one D_i is assigned to true by the assignment J). □ □