Minimising RDF Graphs under Rules and Constraints Revisited [⋆]

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

[1] Technische Universität Wien, {`pichler, skritek, woltran`}`@dbai.tuwien.ac.at`
[2] DERI, National University of Ireland, Galway `axel.polleres@deri.org`

Abstract. Based on practical observations on rule-based inference on RDF data, we study the problem of redundancy elimination in RDF in the presence of rules (in the form of Datalog rules) and constraints (in the form of so-called tuple-generating dependencies). To this end, we investigate the influence of several problem parameters (like restrictions on the size of the rules and/or the constraints) 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.

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 some sense 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,[3] 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] and (2–6) rules for OWL2RL representing inverse properties and property chains:[4]

(1) { `S P O . P` *subPropertyOf* `Q . uri(Q)` } $\Rightarrow$ { `S Q O` }
(2) { `S P O . P` *inverseOf* `Q . blank(O)` $\wedge$ `uri(Q)` } $\Rightarrow$ { `O Q S` }

⋆ 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).

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

[4] 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. $\dots X_n$ *first* P_n . X_n *rest* *nil*. using reserved terms *first*, *rest*, *nil*.

(3) { `S P O . P` *inverseOf* `Q . uri(O)` $\wedge$ `uri(Q)` } $\Rightarrow$ { `O Q S` }
(4) { `P` *inverseOf* `Q . uri(Q)` } $\Rightarrow$ { `Q` *inverseOf* `P` }
(5) { `P` *inverseOf* `Q . blank(Q)` } $\Rightarrow$ { `Q` *inverseOf* `P` }
(6) { `S` $\mathtt{P}_0$ $\mathtt{O}_1$`. ...` $\mathtt{O}_n$ $\mathtt{P}_n$ `O. P` *propertyChainAxiom* ($\mathtt{P}_0$ ...$\mathtt{P}_n$) }$\Rightarrow${ `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) *maker inverseOf made*. *made inverseOf maker*.
(12) *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, statement (9) as well as the statement *maker inverseOf made*. from line (11) 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 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 rule (6) as a constraint, we could say that graph G alone without rules satisfies this constraint, and likewise the closure of G with respect to rules (1)-(3) does.

Note that tgds can be more general than (Horn) rules in that they 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

(13) { 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 (13) holds on the closure of G with respect to rule (1) but – as opposed to the constraint reading of (6) – not on G alone.

The primary goal of our work is a systematic complexity analysis of both graph and rule minimisation under constraints. To this end, we investigate the influence of several problem parameters (like restrictions on the size of the rules and/or the constraints) 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 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).

We shall come up with a collection of complexity results, ranging from tractability to Σ_3^P-completeness. Additionally, we address the orthogonal problem of rule minimisation, which has not been studied so far. 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. We shall also briefly touch on the related problem of reducing rules or triples without preserving completeness of the entire closure, but only ensuring that the answers to certain queries are preserved. For instance, suppose that, in our example, we are interested only in completeness with respect to *creator* statements. Then rules (2)–(5) could in fact be dropped.

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 6. Sections 3–5 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.

• *Problem Variations.* In Section 5, 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 and of requiring only that the answers to a given set of queries must be preserved by the minimisation of the graphs or 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 [12].

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.[5] 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 $\mu : B \rightarrow UBL$ 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 all $\mathtt{B}$ on which h is defined. Given G_1, G_2, deciding whether there exists a homomorphism $G_2 \rightarrow G_1$ (and 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 $\mathtt{uri}(\cdot)$, $\mathtt{blank}(\cdot)$, $\mathtt{literal}(\cdot)$ (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 $\forall \boldsymbol{x}\big(\mathcal{A}nte(\boldsymbol{x}) \rightarrow (\exists \boldsymbol{y})\mathcal{C}on(\boldsymbol{x}, \boldsymbol{y})\big)$ (where $\boldsymbol{y}$ denotes the blank nodes occurring in $\mathcal{C}on$ only, while $\boldsymbol{x}$ 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 $\boldsymbol{x}$ mapping BGP($\mathcal{A}nte$) to G, there exists an extension h' of h to $\boldsymbol{y}$ 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.*

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

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.

We say that a rule or constraint is *b-bounded* if both, antecedent and consequent contain at most b triples. 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 $X \cap \Sigma(G)$, resp. $X \cap \Sigma(\mathcal{R})$, where by $\Sigma(\cdot)$ we denote the signature of a graph or ruleset, that is, the subset of UBL used in G, or $\mathcal{R}$, respectively. Finally, we write $[n]$ to denote the set $\{1, \dots, n\}$.

3 RDF 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 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 $\mathcal{C}$?

Definition 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 minimization 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 graph minimization via entailment allows us to replace a graph G by a subgraph $\bar{G} \subset G$ if $\bar{G} \models G$ holds (see [20]). 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 1).

It is easy to see that the condition $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ in Definition 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 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 1. *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 1.

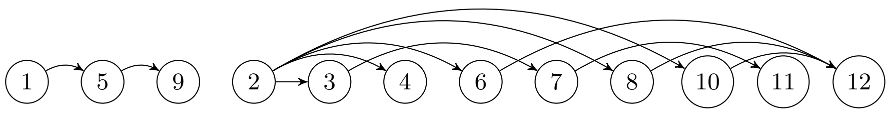

Fig. 1. Dependency graph: Numbers refer to lines in Table 1. An arrow from A to B means that B is a special case of A.

		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. (Where "bb" indicates the set to be b-bounded, and "arb." allows for arbitrary sets.)

Lemma 2. *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.*

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 1, 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 full paper [12].

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. *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 $F = \exists \boldsymbol{x_1} \forall \boldsymbol{y_1} \exists \boldsymbol{x_2} \bigwedge_{i=1}^{n} C_i$, 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 $x_i \in \boldsymbol{x_1}$ 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 $x_i \in \boldsymbol{x_1}$ 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 $\boldsymbol{x_1}$ and $\boldsymbol{y_1}$. Thereby for every valid G' all such homomorphisms define the same truth assignment on $\boldsymbol{x_1}$, hence the values for $\boldsymbol{x_1}$ are determined by the selection of G'. But every homomorphism defines a different truth assignment on $\boldsymbol{y_1}$, and there exists exactly one homomorphism for each of the $2^{|\boldsymbol{y_1}|}$ truth assignments on $\boldsymbol{y_1}$. 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 $\boldsymbol{x_2}$ such that the assignment on $\boldsymbol{x_1}$, $\boldsymbol{y_1}$ and $\boldsymbol{x_2}$ 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. $\square$

Lemma 4. *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, we concentrate on MINI-RDF$^{\subseteq}$ and prove its NP-hardness by reduction from the 3-SAT problem. First, we fix the 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 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 $G = \{l^*_{i,j}\ \mathit{in}\ c_i \mid i \in [n], j \in [3]\} \cup \{l^*_{i,j}\ \mathit{active}\ c_i \mid i \in [n], j \in [3]\} \cup \{x_j\ \mathit{clash}\ \bar{x}_j \mid x_j \text{ in } F\}$, 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. $\square$

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. [15]). 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 3. *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. *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 2. *Deciding* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *for a set $\mathcal{R}$ of b-bounded rules (for fixed b) is* NP*-complete.*

Proof. The hardness is shown by reduction from the 3-SAT problem. The idea of the reduction is, given some formula F in 3-CNF, to provide in the graph G an encoding of all combinations of truth values under which a clause evaluates to true, and to design the set $\mathcal{R}$ of rules such that the triples derivable by $\mathcal{R}$ encode which literals occur together in some clause in F. Further, $\mathcal{R}$ is chosen such that $Cl_{\mathcal{R}'}(G) \models Cl_{\mathcal{R}}(G)$ holds for any subset $\mathcal{R}' \subset \mathcal{R}$ iff *all* triples derivable by $\mathcal{R}$ can be mapped into G. This mapping then defines a valid satisfying truth assignment for F. □

Theorem 3. *The problem* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$*, for a b-bounded set $\mathcal{R}$ of rules, can be decided in* PTIME*, while for arbitrary rules $\mathcal{R}$, both,* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ *and* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *are in Δ_2^P.*

Proof. In the b-bounded case, the closure can be computed efficiently: 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. For arbitrary rules, the same strategy can be used, but now an NP-oracle is needed to test if a rule is applicable. For RDF-RULEMIN$^{\models}$, this oracle can then also be used to test for entailment. □

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 [16]). 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 [17]), 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 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 5. *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 4. *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 $K + k$. $\square$

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 5. *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 1. Hardness is shown by a modification of the reduction given in the proof of Lemma 3. 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 $v_i \in \boldsymbol{x_1}$, 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 $v_i \in \boldsymbol{x_1}$ (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. $\square$

From the (full) proof of Lemma 4, 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. *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 1. *Hardness* is shown by reduction from the SAT problem. The reduction is very similar to the one of Lemma 4, 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.

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 $G = \{l^*_{i,j}\ in\ c_i \mid i \in [n], j \in [k_i]\} \cup \{l^*_{i,j}\ active\ c_i \mid i \in [n], j \in [k_i]\} \cup \{x_j\ clash\ \bar{x}_j \mid x_j \text{ in } F\} \cup \{c_i\ clause\ c_i \mid i \in [n]\}$ and $\mathcal{C} = \{\ \{X\ active\ I\,.\,X\ in\ J\} \Rightarrow \{X\ active\ J\}; \{X\ clash\ X'\,.\,X\ active\ I\,.\,X'\ active\ I'\,.\,Y\ in\ J\} \Rightarrow \{Y\ active\ J\}; \{I\ clause\ I\} \Rightarrow \{X\ active\ I\}\} \cup \{\{A\ B\ C\} \Rightarrow \{c_i\ clause\ c_i\} \mid i \in [n]\} \cup \{\{A\ B\ C\} \Rightarrow \{l^*_{i,j}\ in\ c_i\} \mid i \in [n], j \in [k_i]\} \cup \{\{A\ B\ C\} \Rightarrow \{x_j\ clash\ \bar{x}_j\} \mid x_j \text{ in } F\}$ where $l^*_{i,j} = x_\alpha$ if $l_{i,j} = x_\alpha$ and $l^*_{i,j} = \bar{x}_\alpha$ if $l_{i,j} = \neg x_\alpha$. $\square$

Note that the above proof would work – by a grounding of the last constraints and introducing some additional constraints – as well if no blank nodes in the predicate positions were allowed. However, the proof would be less compact.

We now show that the complexity of the 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: (1) 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. (2) 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)–(5) in Section 1, where the filter conditions guaranteed valid intermediate triples.

We conclude this section by discussing a further variant of (graph or rule) minimisation that guarantees completeness only w.r.t. a given set of conjunctive queries (CQs). Such a minimisation is of high interest, e.g. when importing data into an RDF Store that provides a narrow query interface only. Below we give a formal definition for the CQ-variant of the MINI-RDF$^{\subseteq}$ problem.

Definition 6. 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, defined by BGPs.
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 $\mathcal{C}$?

The CQ-variants of MINI-RDF$^{\models}$, RDF-RULEMIN$^{\subseteq}$, and RDF-RULEMIN$^{\models}$ are defined analogously. A detailed complexity analysis of the CQ-variants of all problems studied here is outside the scope of this paper. However, we briefly mention that the 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 $\mathcal{Q} = \{S\ P\ O\}$.

6 Conclusions

We proved a collection of complexity results for minimisation problems over RDF graphs where we considered various restrictions on the rules and tdgs. 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 [18] showed that such a restriction decreases complexity.

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 [19].

We have briefly discussed further variants of minimisations like (graph or rule) minimisations that guarantee completeness with respect to a given set of conjunctive queries (CQs). We have sketched that the hardness results from Sections 3 and 4 still hold when we take CQs into account. Identifying the precise complexity of RDF minimisation relative to a set of CQs is an important task for future work. As another direction of future work, 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 Rec.
9. Hayes, P.: RDF semantics. Technical report, W3C (February 2004) W3C Rec.
10. Lausen, G., 0002, M.M., Schmidt, M.: Sparqling constraints for rdf. In: Proc. EDBT'08. Volume 261 of ACM International Conference Proceeding Series., ACM (2008) 499–509
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 (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. 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
16. 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
17. 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
18. Vardi, M.: On the complexity of bounded-variable queries. In: Proc. PODS'95, ACM Press (1995) 266–276
19. de Bruijn, J.: RIF RDF and OWL Compatibility (October 2008) W3C Cand. Rec.
20. Gutierrez, C., Hurtado, C., Mendelzon, A.: Foundations of semantic web databases. In: Proc. PODS'04, ACM (2004) 95–106