Knowledge-Base Degrees of Inconsistency: Complexity and Counting

Johannes K. Fichte[1] **Markus Hecher**[2] **Arne Meier**[3]
[1] TU Dresden
[2] TU Wien
[3] Leibniz Universität Hannover
johannes.fichte@tu-dresden.de, markus.hecher@tuwien.ac.at, meier@thi.uni-hannover.de

Abstract

Description logics (DLs) are knowledge representation languages that are used in the field of artificial intelligence (AI). A common technique is to query DL knowledge-bases, e.g., by Boolean Datalog queries, and ask for entailment. But real world knowledge-bases often have a certain inconsistency (with respect to a given query) or we are required to estimate a degree of inconsistency when using a knowledge-base. In this paper, we provide a complexity analysis of fixed-domain non-entailment (NE) on Datalog programs for well-established families of knowledge-bases (KBs). We exhibit a detailed complexity map for the decision cases, counting and projected counting, which may serve as a quantitative measure for inconsistency of a KB with respect to a query. Our results show that NE is natural for the second, third, and fourth level of the polynomial (counting) hierarchy depending on the type of the studied query (stratified, tight, normal, disjunctive) and one level higher for the projected versions. Further, we show fixed-parameter tractability by bounding the treewidth, provide a constructive algorithm, and show its theoretical limitation in terms of conditional lower bounds.

1 Introduction

Description logics (DL) have proven itself as a central framework in the field of artificial intelligence (AI) (Krötzsch, Simancik, and Horrocks 2012; Baader et al. 2017). Since DL was introduced by Brachman and Levesques in 1984, the formalism has become a host for expressing knowledge representation problems and reasoning about the relevant concepts of an application domain (ontologies). As the application domain is quite vast (Hitzler et al. 2012; Knublauch, Musen, and Rector 2004; Eiter et al. 2015; Klinov 2008), a plethora of different extensions as well as restrictions have emerged (Calvanese 2006; Horrocks, Kutz, and Sattler 2006; Horrocks and Sattler 2007). The versatility of DL is one of its strength and, as a result, new extensions or combinations with other concepts or formalisms are common (Baader, Küsters, and Wolter 2003; Baader et al. 2005; Motik 2006). In general, a knowledge-base consists of different components: an ABox, a TBox, and a RBox. An *ABox* (*assertion box*) can be seen as a prescribed part of the model, a *TBox* (*terminology box*) is a set of so-called inclusion-axioms that have to be obeyed in every world of the model, and an *RBox* (*role box*) is a set of relation constraints. Reasoning in DLs often involves querying knowledge-bases which, when asking for closed-world, can be achieved by Boolean Datalog programs. While constructing those queries seem straight-forward, huge real world databases often contain data entries that are inconsistent with respect to a considered query. In other words, given a knowledge-base and a Datalog query, a model of a knowledge-base cannot consistently be extended to the query and is seen to be inconsistent with respect to it. We want to estimate a degree of inconsistency by counting inconsistencies. This can be particularly interesting in a setting where we need a quantitative measure on the models that cannot be entailed. This is a stronger notion of reasoning than classical skeptical reasoning (asking for containment in all models). Let us consider the following example from the propositional setting to illustrate this approach. Note that *stable* models are defined in terms of subset-minimality. We give a more elaborate example in the sequel of the paper.

Example 1. *Consider the following entailment question:* $\{p(a) \vee p(b)\} \models \{p(c) \leftarrow; \leftarrow p(a), p(b)\}$*? The models of* $\{p(a) \vee p(b)\}$ *are* $\{p(a)\}$*,* $\{p(b)\}$*, and* $\{p(a), p(b)\}$*. However, only* $\{p(a), p(c)\}$ *and* $\{p(b), p(c)\}$ *are stable models of the program* $\{p(c) \leftarrow; \leftarrow p(a), p(b)\}$ *extending* $\{p(a)\}$ *and* $\{p(b)\}$*, respectively. As a result, the last of the three models cannot be extended to a stable model.* ⊣

In this paper, we investigate the complexity of fixed-domain *non-entailment (NE)* on Datalog programs for $\mathcal{SROIQ}$ or $\mathsf{DL}_{\mathsf{min}}$ knowledge-bases, which may serve as a quantitative measure for inconsistency. The DL $\mathcal{SROIQ}$ is quite expressive: it allows to express transitivity, (ir)reflexivity, antisymmetry, subrole-relation, inverse, disjointness of roles, as well as negated role assertions, complex role inclusions, the universal role, and local reflexivity (Horrocks, Kutz, and Sattler 2006). The DL $\mathsf{DL}_{\mathsf{min}}$ is, as already stated by its name, quite restricted: only TBox-axioms of the form $A \sqsupseteq \neg B$ as well as atomic assertions $A(a), r(a, b)$ are allowed, with A, B are concept names, r a role name and a, b individual names. In our setting, we also incorporate a more fine-grained analysis by considering the treewidth, which is a popular parameter that makes various NP-hard problems tractable and is widely used in the field of artificial intelligence (Gottlob and Szeider 2007;

Name	Syntax	Semantics
inverse role	r^-	$\{ (x,y) \in \Delta \times \Delta \mid (y,x) \in r^{\mathcal{I}} \}$
universal role	u	$\Delta \times \Delta$
top	$\top$	Δ
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
nominals	$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$
univ. restr.	$\forall s.C$	$\{x \mid \forall y.(x,y) \in s^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}} \}$
exist. restr.	$\exists s.C$	$\{x \mid \exists .(x,y) \in s^{\mathcal{I}} \wedge y \in C^{\mathcal{I}} \}$
self concept	$\exists s.$self	$\{x \mid (x,x) \in s^{\mathcal{I}} \}$
qualified no.	$\leq nr.C$	$\{x \mid \#\{y \in C^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \leq n\}$
restriction	$\geq nr.C$	$\{x \mid \#\{y \in C^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \geq n\}$

Sets	Axiom α	$\mathcal{I}$ satisfies α if
ABox $\mathcal{A}$	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
	$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$
	$a \equiv b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
	$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$
RBox $\mathcal{R}$	$r_1 \circ \ldots \circ r_n \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \ldots \circ r_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$
	$\mathsf{Dis}(r_1, r_2)$	$r_1^{\mathcal{I}} \cap r_2^{\mathcal{I}} = \emptyset$
TBox $\mathcal{T}$	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

Table 1: Syntax and semantics of $\mathcal{SROIQ}$, where a, b, a_1, …, a_n denote individual names, s is a role name, r, r_1, …, r_n are role expressions, and C and D concept expressions.

Gottlob, Pichler, and Wei 2006, 2007) and databases (Grohe 2007). Our main contributions are as follows: (I) We establish the *computational complexity* for deciding NE and for (projected) counting of NE anti-witnesses. Our results show that NE is complete for the *second*, *third*, and *fourth level* of the *polynomial (counting) hierarchy* depending on the type of the studied query (stratified, tight or normal, disjunctive) and increases by one level when considering projected answers in the setting of bounded domains. (II) We provide a novel graph (*knowledge-program graph*) to define treewidth on non-ground Datalog programs and show fixed-parameter tractability by bounding the treewidth of the knowledge-program graph and bounding the domain. (III) We establish dynamic programming algorithms for solving the problems and show also matching conditional lower bounds.

Related Works Parameterized complexity results for description logics, however, mostly hardness results, have recently been established by de Haan (2018). Treewidth is widely used for fine-grained complexity analyzes and to establish algorithms that provide tractability when bounding the treewidth, e.g., artificial intelligence (Gottlob and Szeider 2007), knowledge representation (Gottlob, Pichler, and Wei 2006), abduction in Datalog (Gottlob, Pichler, and Wei 2007), and databases (Grohe 2007). In the context of counting and projected counting treewidth has been considered in various areas (Fichte, Hecher, and Meier 2018; Fichte et al. 2017; Fichte and Hecher 2019; Fichte, Hecher, and Schindler 2018; Fichte et al. 2018a). Also competitive implementations are available (Fichte et al. 2018b, 2020; Hecher, Thier, and Woltran 2020; Dudek, Phan, and Vardi 2020). Conditional lower bounds have been considered by Pan and Vardi (2006) and Fichte, Hecher, and Pfandler (2020). Inconsistencies and distance measures, in terms of the number of formulas responsible for an inconsistency and the propositions in the language affected by the inconsistency, have been considered by Ma, Qi, and Hitzler (2010; 2011). Para-consistent semantics for handling inconsistencies occurring in the combination of description logics and rules have been suggested by Huang, Li, and Hitzler (2012). Repair-related problems have been considered in databases, but are conceptually different from our setting, as databases usually only assume closed world. Eiter et al. (2008) studied the combination of ASP and DL in terms of fixpoint characterizations, strong and a weak answer set semantics, and computational complexity (EXP, NEXP, co-EXP, P^{NEXP}). Gaggl, Rudolph, and Schweizer (2016) considered the complexity for standard reasoning and query answering under the fixed-domain semantics for DLs. While they investigate the complexity under queries that implement stable model semantics, their work is limited to the ground case and does not incorporate disjunctive queries. In contrast to our work, they consider a different formalism restricting the occurrences of atoms in the heads in the rules of the query. They do not consider treewidth. We state explicit runtime bounds yielding tractability and precise runtime guarantees. Rosati (2011) considered the complexity of inconsistency by minimally repairing an ABox while keeping the TBox untouched. There are various works that consider treewidth or other parameterizations in the realm of DL (Bienvenu et al. 2013; Simančík, Motik, and Horrocks 2014; Barceló et al. 2019; Bienvenu et al. 2017). We consider the fixed-domain setting, which in practical solving is a fairly weak limitation, but allows for much better complexity behavior. In contrast to the mentioned works above, we establish exact runtimes that cannot be significantly improved under ETH (both upper and lower bounds). Projected model counting is also emerging topic in practical experimental evaluations (Fichte, Hecher, and Hamiti 2020).

2 Preliminaries

Let $\mathsf{tower}(i, p)$ be $\mathsf{tower}(i-1, 2^p)$ if $i > 0$ and p otherwise.

Description Logics For an introduction to the description logic $\mathcal{SROIQ}$, we refer to its original source (Horrocks and Sattler 2007). We briefly outline its syntax and semantics. Let Δ be a finite, fixed domain. Let $\mathsf{N_I}$, $\mathsf{N_C}$, and $\mathsf{N_R}$ be finite disjoint sets. We call $\mathsf{N_I}$ *individual names*, $\mathsf{N_C}$ *concept names*, and $\mathsf{N_R}$ *role names*, which can be used to form complex expressions as displayed in Table 1. An axiom in description logic is a logical statement that relates roles and concepts. Table 1 provides the axioms that are allowed for the description logic $\mathcal{SROIQ}$ and its *ABox* (*assertion box*), *TBox* (*terminology box*), and *RBox* (*role box*). Intuitively, an ABox contains sentences stating where in the hierarchy, individuals belong (i.e., relations between individuals and concepts), and a TBox contains sentences describing concept hierarchies (i.e., relations between concepts), and an RBox the relations of constraints. Let $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ be a tuple called *knowledge-base*, where $\mathcal{A}$ is an ABox, $\mathcal{T}$ is

a TBox, and $\mathcal{R}$ is an RBox. Then, we abbreviate the individual names, concept names, and role names that occur in $\mathcal{K}$ by $\mathsf{N_I}(\mathcal{K})$, $\mathsf{N_C}(\mathcal{K})$, and $\mathsf{N_R}(\mathcal{K})$, respectively. Further, we let $\mathsf{N}(\mathcal{K}) := \mathsf{N_I}(\mathcal{K}) \cup \mathsf{N_C}(\mathcal{K}) \cup \mathsf{N_R}(\mathcal{K})$ be the names occurring in $\mathcal{K}$ and we let these definitions naturally extend to ABox, TBox, RBox, and axioms. An *interpretation* $\mathcal{I}$ is a tuple $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ where $\cdot^I$ is a function that maps elements in N_I to elements in Δ, elements in N_C to elements in 2^Δ, elements in N_R to elements in $2^{\Delta\times\Delta}$, and for extended concepts to complex role and concept expressions according to Table 1. Alternatively, we may view an interpretation $\mathcal{I}$ as a triple $\langle I, C, R\rangle$ with $I\colon \mathsf{N_I}(\mathcal{K}) \to \Delta$, $C\colon \mathsf{N_C}(\mathcal{K}) \to 2^\Delta$, and $R\colon \mathsf{N_R}(\mathcal{K}) \to 2^{\Delta\times\Delta}$. We say that $\mathcal{I}$ satisfies $\mathcal{K}$, or $\mathcal{I} \models \mathcal{K}$ for short and call $\mathcal{I}$ *model* of $\mathcal{K}$, if it satisfies all axioms of $\mathcal{A}$, $\mathcal{T}$, and $\mathcal{R}$ where the satisfaction of axioms is given as in Table 1. We say that $\mathcal{K}$ entails an axiom α, or $\mathcal{K} \models \alpha$ for short, if all models of $\mathcal{K}$ are models of α.

The DL $\mathsf{DL_{min}}$ allows only TBox-axioms of the form $A \sqsupseteq \neg B$ as well as atomic assertions $A(a), r(a, b)$, with A, B are concept names, r a role name and a, b individual names.

Datalog Programs Let Δ be a fixed, finite domain of cardinality at least 2, and $\mathsf{N_P}$ be a set of *predicate names*. An *atom* is an expression $p(t_1, \ldots, t_c)$, where a predicate $p \in \mathsf{N_P}$ is of bounded-arity $c \geq 0$ and each t_i is either a (first-order) variable or an element of domain Δ. We say an atom is *Boolean* if it contains only domain elements. We usually write upper case letters like X or X attributed with a subscript for a variable. A *(disjunctive) rule* r is an expressions of the form $a_1 \vee \cdots \vee a_\ell \leftarrow a_{\ell+1}, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n.$ with $n > 0$, $\ell \leq m \leq n$ and where $a_1, \ldots, a_m$ are atoms. We let $H_r := \{a_1, \ldots, a_\ell\}$, $B_r^+ := \{a_{\ell+1}, \ldots, a_m\}$, and $B_r^- := \{a_{m+1}, \ldots, a_n\}$. A *ground rule* is of the same form as a disjunctive role, however, $a_1, \ldots, a_m$ are Boolean atoms. A rule r is *normal* if $|H_r| \leq 1$. We say that a program has a certain property if all its rules have the property.

A *program* is a finite set of rules. We denote the sets of *atoms* occurring in a rule r, resp., in a program $\mathcal{P}$, by $\mathrm{at}(r) := H_r \cup B_r^+ \cup B_r^-$, resp., by $\mathrm{at}(\mathcal{P}) := \bigcup_{r\in\mathcal{P}} \mathrm{at}(r)$. Further, the predicates of $\mathsf{N_P}$ occurring in a rule r or program $\mathcal{P}$ are addressed by $\mathsf{N_P}(r)$ or $\mathsf{N_P}(\mathcal{P})$, respectively. A substitution σ is a total mapping from a given set of variables to Δ. Then, $\mathrm{Ground}(\mathcal{P})$ is the set of ground rules $r\sigma$ obtained by applying, to each rule $r \in \mathcal{P}$, all possible substitutions σ from variables in $\mathcal{P}$ to Δ.

A normal program $\mathcal{P}$ is *stratified* if there is a mapping $\mathrm{str}\colon \mathrm{at}(\mathrm{Ground}(\mathcal{P})) \to \mathbb{N}$ such that for each rule $r \in \mathrm{Ground}(\mathcal{P})$ the following is true: (i) if $x \in H_r$ and $y \in B_r^+$, then $\mathrm{str}(x) \leq \mathrm{str}(y)$, and (ii) if $x \in H_r$ and $y \in B_r^-$, then $\mathrm{str}(x) < \mathrm{str}(y)$. We say that a normal program is *tight* if it has no cycles w.r.t. the graph that has as vertices the Boolean atoms $\mathrm{at}(\mathrm{Ground}(\mathcal{P}))$ and a directed edge (x, y) between any two atoms x,$y \in \mathrm{at}(\mathrm{Ground}(\mathcal{P}))$ for which there is a rule $r \in \mathrm{Ground}(\mathcal{P})$ with $x \in H_r$ and $y \in B_r^+$. We denote the class of all normal programs by **Normal**, the class of all stratified programs by **Strat**, the class of all tight programs by **Tight**, and the class of all programs by **Disj**.

An *interpretation* $\mathcal{J}$ is a set of *Boolean* atoms. Then, $\mathcal{J}$ *satisfies* a ground rule r if $(H_r \cup B_r^-) \cap \mathcal{J} \neq \emptyset$ or $B_r^+ \setminus \mathcal{J} \neq \emptyset$. $\mathcal{J}$ is a *model* of $\mathcal{P}$ if it satisfies every ground rule $r \in \mathrm{Ground}(\mathcal{P})$, in symbols $\mathcal{J} \models \mathcal{P}$. The *GL-reduct* of $\mathcal{P}$ (Gelfond and Lifschitz 1991) under $\mathcal{J}$ is the program $\mathcal{P}^\mathcal{J} = \{\, H_r \leftarrow B_r^+ \mid \mathcal{J} \cap B_r^- = \emptyset, r \in \mathrm{Ground}(\mathcal{P}) \,\}$. We say that $\mathcal{J}$ is a *stable model* of a program $\mathcal{P}$ if $\mathcal{J}$ is a subset-minimal model of $\mathcal{P}^\mathcal{J}$. For tight programs this is equivalent (Lin and Zhao 2003) to $\mathcal{J} \models \mathcal{P}$ such that every $a \in \mathcal{J}$ is *justified (by $\mathcal{P}$)*, i.e., there has to exist $r \in \mathrm{Ground}(\mathcal{P})$ with $a \in H_r$, $\mathcal{J} \subseteq B_r^+$ and $\mathcal{J} \cap B_r^- = \emptyset$. Deciding whether a ground program has a stable model (*consistency problem*) is Σ_2^P-complete (Eiter et al. 2007; Eiter and Gottlob 1995). Deciding whether a program has a stable model (*consistency problem of a program with bounded-arities*) as well as deciding whether an atom is in a stable model (*brave reasoning*) is Σ_3^P-complete (Eiter et al. 2007).

Tree Decompositions Let $G = (V, E)$ be a graph. A *tree decomposition (TD)* of a graph G is a pair $\mathcal{T} = (T, \chi)$, where T is a rooted tree and χ a mapping that assigns to each node t of T a set $\chi(t) \subseteq V$, called a *bag*, such that the following conditions hold: (i) $E \subseteq \cup_{t \text{ of } T}\{\, \{u, v\} \mid u, v \in \chi(t) \,\}$; and (ii) for each r, s, t, such that s lies on the path from r to t, we have $\chi(r) \cap \chi(t) \subseteq \chi(s)$. Then, the *width of $\mathcal{T}$* corresponds to $\max_{t \text{ of } T} |\chi(t)| - 1$ and the *treewidth* is the minimum width over all TDs of G. In order to simplify case distinctions in the algorithms, *we assume nice TDs* as follows: For a node t of T, we say that $\mathrm{type}(t)$ is *leaf* if t has no child nodes and $\chi(t) = \emptyset$; *join* if t has exactly two child nodes t_1, t_2 with $\chi(t) = \chi(t_1) = \chi(t_2)$; *intr* if t has only one child node t_1 with $\chi(t_1) \subseteq \chi(t)$ and $|\chi(t)| = |\chi(t_1)| + 1$; *forget* if t has only one child node t_1 s.t. $\chi(t_1) \supseteq \chi(t)$ and $|\chi(t_1)| = |\chi(t)| + 1$. If for every node $t \in N$, $\mathrm{type}(t) \in \{\textit{leaf}, \textit{join}, \textit{intr}, \textit{forget}\}$ and the bag of the root is empty, then the TD is called *nice*.

3 Datalog Queries in Description Logics

We use fixed-domain semantics inspired by earlier work (Gaggl, Rudolph, and Schweizer 2016). However, we extend the notion and enable full Datalog programs. Given a knowledge-base $\mathcal{K}$, a fixed domain Δ, and a Datalog program $\mathcal{P}$, we say that $\mathcal{K} \models\!\!\!\sim \mathcal{P}$ if and only if every model $\mathcal{I} = (\Delta, \cdot^\mathcal{I})$ of $\mathcal{K}$ can be extended to a stable model $\mathcal{J}$ of $\mathcal{P}$. More precisely, $\mathcal{K} \models\!\!\!\sim \mathcal{P}$ is true if and only if there a subset-minimal model $\mathcal{J}$ of $(\mathrm{Ground}(\mathcal{P}) \cup \mathcal{P}_\mathcal{I})^\mathcal{J}$ where $\mathcal{P}_\mathcal{I} := \{\, C(d) \leftarrow \mid d \in C^\mathcal{I}, C \in \mathsf{N_C}(\mathcal{K}) \,\} \cup \{\, R(d_1, d_2) \leftarrow \mid (d_1, d_2) \in R^\mathcal{I}, R \in \mathsf{N_R}(\mathcal{K}) \,\}$. If $\mathcal{I}$ cannot be extended to a stable model $\mathcal{J}$ of $\mathcal{P}$, we say $\mathcal{I}$ is an *anti-witness of* $\mathcal{K} \models\!\!\!\sim \mathcal{P}$.

We consider the following fixed-domain non-entailment problem on specific bounded-arity Datalog programs. Let $\mathrm{NE}_\mathcal{C}$; $\mathcal{C} \in \{\textbf{Strat}, \textbf{Tight}, \textbf{Normal}, \textbf{Disj}\}$ be the problem asking, given KB $\mathcal{K}$ and a $\mathcal{C}$-program $\mathcal{P}$, $\mathcal{K} \not\models\!\!\!\sim \mathcal{P}$? Sometimes we write NE for $\mathrm{NE}_{\textbf{Disj}}$. The following example illustrates the problem NE.

Example 2. *Consider the following knowledge-base $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \emptyset)$, which describes a "medical" setting and relations between diseases and symptoms. First, take as the domain $\Delta = \{\mathrm{donald}, \mathrm{emanuel}, \mathrm{silvio}\}$ describing three people and define the ABox $\mathcal{A}$ and the TBox $\mathcal{T}$ as follows: $\mathcal{A} := \{\mathsf{Meet}(P_1, P_2), \mathsf{Contagious}(P_1), \mathsf{Meet}(P_2, P_3), P_1 \neq P_2, P_2 \neq P_3, P_1 \neq P_3\}$, $\mathcal{T} := \{$*

$\leq$ 1u.Mask, Contagious $\equiv$ Flu $\sqcup$ Sinusitis, Exposed $\equiv$ $\exists$Meet.(Contagious $\sqcap \neg$Mask), Flu $\equiv$ Fever $\sqcap$ Exposed, Sinusitis $\equiv$ $\neg$Fever $\sqcap$ Exposed, Meet $\circ$ Meet $\equiv$ Meet, Meet $\equiv$ Meet$^-$}. *Then, we ask the following query* $\mathcal{P} := \{\leftarrow \neg\mathsf{superspread};\ \mathsf{superspread} \leftarrow \mathsf{Contagious}(X), \mathsf{Contagious}(Y), X \neq Y\}$*, where we simply consider "$\neq$" as a shortcut, since we have a fixed domain. When we ask for entailment of* $\mathcal{K} \models \mathcal{P}$*,* $\mathcal{P}$ *expresses that there are at least two contagious people (superspread). Observe that this does not hold, i.e.,* $\mathcal{P}$ *is a positive instance for* NE*. However, the number of anti-witnesses to* $\mathcal{K} \models \mathcal{P}$ *is small compared to all models* $\mathcal{I}$ *of* $\mathcal{K}$ *with* $\mathcal{I} \models \mathcal{P}$*. This is justified since only interpretations* $\mathcal{J}$ *with* $P_1^{\mathcal{J}} \in \mathsf{Mask}^{\mathcal{J}}$ *are anti-witnesses, attributing to about* $\frac{1}{3}$ *of the models of* $\mathcal{K}$*. Hence, weaker forms of entailment are still rather likely.* $\dashv$

Theorem 3. *The combined complexity of* $\mathrm{NE}_{\textbf{\textit{Strat}}}$ *w.r.t. a* $\mathcal{SROIQ}$ *or* $\mathsf{DL}_{\mathsf{min}}$ *knowledge-base is* Σ_2^{P}*-complete.*

Proof (Sketch). We reduce $\mathsf{QBFVAL}_{\forall,2}$ to the complement of $\mathrm{NE}_{\textbf{Strat}}$. Let $\Phi = \forall p_1, \ldots, p_k \exists q_1, \ldots, q_m \varphi$ with φ in 3CNF, that is $\varphi = \bigwedge_{i=1}^{n} C_i$ and $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ for $1 \leq i \leq n$, $k, m, n \in \mathbb{N}$, and $\ell_{i,j}$ are literals from variables in $V := P \cup Q$, where $P := \{p_1, \ldots, p_k\}$, $Q := \{q_1, \ldots, q_m\}$. Next, we define a $\mathsf{DL}_{\mathsf{min}}$ knowledge-base $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \emptyset)$, a fixed-domain Δ, and a non-ground stratified Datalog program $\mathcal{P}$ such that $\mathcal{K} \not\models \mathcal{P}$ if and only if Φ is valid. Set $\Delta := \{0, 1\}$ and define $\mathcal{K}$ as follows. Let the ABox $\mathcal{A}$ contain the following axioms:

$$\mathsf{Select}_v(x_v), \mathsf{UnSelect}_v(y_v) \text{ for all } v \in \{p_1, \ldots, p_k\}$$

Let the TBox $\mathcal{T}$ contain the following axioms:

$$\mathsf{Select}_v \sqsubseteq \neg\mathsf{UnSelect}_v \text{ for all } v \in \{p_1, \ldots, p_k\}$$

Fix an arbitrary total ordering $<$ on the variables of Φ. For a set W of variables and a clause or term T, let $\bar{W}(T) = (w_1, \ldots, w_e)$ be the ordered sequence of variables of T that are also in W where $0 \leq e \leq 3$, i.e., $\{w_1, \ldots, w_e\} = W \cap \mathrm{Vars}(T)$. We define program $\mathcal{P}$ as the union of programs

$$\begin{aligned}
&\textstyle\bigcup_{i=1}^{n} \{\mathsf{sat}_{C_i}(\theta) \leftarrow \widetilde{\ell}_{i,j} \mid \ell_{i,j} \notin \mathrm{Lit}(Q), \theta\colon \bar{Q}(C_i) \to \{0,1\}, \\
&\qquad\qquad |\bar{Q}(C_i)| < 3, \theta \not\models C_i \cap \mathrm{Lit}(Q)\}, \\
&\textstyle\bigcup_{i=1}^{n} \{\mathsf{sat}_{C_i}(\theta) \mid \theta\colon \bar{Q}(C_i) \to \{0,1\}, \theta \models C_i\}, \\
&\qquad \{\mathsf{sat} \leftarrow \mathsf{sat}_{C_1}(\bar{Q}(C_1)), \ldots, \mathsf{sat}_{C_n}(\bar{Q}(C_n));\ \leftarrow \neg\mathsf{sat}\}
\end{aligned}$$

where $\widetilde{\ell} := \mathsf{Select}_v(0)$ if $\ell = \neg v$ and $\widetilde{\ell} := \mathsf{Select}_v(1)$ if $\ell = v$. Note that θ is interpreted as bit-vector, where expressions of the form "$\mathsf{sat}_{C_i}(\bar{Q}(C_i))$" contain for "$\bar{Q}(C_i)$" tuples of FO-variables. The construction with the tuples $\bar{Q}$ resembles an idea influenced by Eiter et al. (2007, Lemma 6). When asking for entailment, program $\mathcal{P}$ ensures that only those models of $\mathcal{K}$ are also stable models of Φ for which we have validity (i.e., for all $\{p_1, \ldots, p_k\}$ we can satisfy all clauses by setting $\{q_1, \ldots, q_m\}$ using FO-variables $\bar{Q}(C_i)$ of sat_{C_i}.

It remains to argue membership. We can obtain fixed-domain non-entailment of a query $\mathcal{P}$ from a $\mathcal{SROIQ}$ knowledge-base $\mathcal{K}$ ($\mathcal{I} \not\models \mathcal{P}$) by (a) guessing an interpretation $\mathcal{I}$ and verifying whether $\mathcal{I}$ is a model of $\mathcal{K}$ and then (b) by checking whether $\mathcal{P} \cup \mathcal{P}_{\mathcal{I}}$ has a stable model. Since for (a) we can simply guess an interpretation and checking fixed-domain satisfiability of a $\mathcal{SROIQ}$ knowledge-base is in NP (Gaggl, Rudolph, and Schweizer 2016, Lem 5) and for (b) consistency of a bounded-arity Datalog query is in Δ_2^{P} (Eiter et al. 2007, Lem. 2), we have a machine witnessing membership in $\mathrm{NP}^{\Delta_2^{\mathrm{P}}} = \mathrm{NP}^{\mathrm{NP}}$. $\square$

Example 4. *Let* $\Phi = \forall p \exists q \varphi$ *and* $\varphi = C_1 \wedge C_2$*, with* $C_1 = (p \vee q), C_2 = (p \vee \neg q)$*. Then,* $\mathcal{A} = \{\mathsf{Select}_p(x_p), \mathsf{UnSelect}_p(y_p)\}$*, and* $\mathcal{T} = \{\mathsf{Select}_p \sqsubseteq \neg\mathsf{UnSelect}_p\}$*. We choose the ordering* $p < q$*. (1)* $\mathcal{P}$ *contains '*$\mathsf{sat}_{C_1}(0) \leftarrow \mathsf{Select}_p(1)$*' and '*$\mathsf{sat}_{C_2}(1) \leftarrow \mathsf{Select}_p(1)$*' as* C_1 *is only satisfied w.r.t. the variable* q *by setting it to 1 and for* C_2 *by setting* q *to* 0*. (2)* $\mathcal{P}$ *contains '*$\mathsf{sat}_{C_1}(1)$*' and '*$\mathsf{sat}_{C_2}(0)$*' as argued before. (3)* $\mathcal{P}$ *contains '*$\mathsf{sat} \leftarrow \mathsf{sat}_{C_1}(X_q), \mathsf{sat}_{C_2}(X_q)$*' as well as '*$\leftarrow \neg\mathsf{sat}$*'. Clearly,* Φ *is false, and, as a result, we have* $\mathcal{K} \not\models \mathcal{P}$*. Then, we construct an interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ *such that* $\mathcal{I} \models \mathcal{K}$ *and it cannot be extended to a stable model* $\mathcal{J}$ *of* $\mathcal{P}$*. Set* $x_p^{\mathcal{I}} = 0$ *and* $y_p^{\mathcal{I}} = 1$*,* $\mathsf{Select}_p^{\mathcal{I}} = \{0\}$*,* $\mathsf{UnSelect}_p^{\mathcal{I}} = \{1\}$*. Now, no stable model* $\mathcal{J}$ *of* $\mathcal{I}$ *can set* sat_{C_1} *and* sat_{C_2} *to contain the same element. As a result,* $\mathsf{sat}^{\mathcal{J}} = 0$*, contradicting* $\leftarrow \neg\mathsf{sat}$*.* $\dashv$

Theorem 5. *The combined complexity of* $\mathrm{NE}_{\textbf{\textit{Disj}}}$ *w.r.t. a* $\mathcal{SROIQ}$ *or* $\mathsf{DL}_{\mathsf{min}}$ *knowledge-base is* Σ_4^{P}*-complete.*

Proof (Sketch). We reduce $\mathsf{QBFVAL}_{\forall,4}$ to the complement of $\mathrm{NE}_{\textbf{Disj}}$. Let $k, m, h, o \in \mathbb{N}$, $P := \{p_1, \ldots, p_k\}$, $Q := \{q_1, \ldots, q_m\}$, $R := \{r_1, \ldots, r_h\}$, and $S := \{s_1, \ldots, s_o\}$. Let $\Phi = \forall p_1, \ldots, p_k \exists q_1, \ldots, q_m \forall r_1, \ldots, r_h \exists s_1, \ldots, s_o \varphi$ with φ in 3CNF, that is $\varphi = \bigvee_{i=1}^{n} C_i$ and $C_i = \ell_{i,1} \wedge \ell_{i,2} \wedge \ell_{i,3}$ for $1 \leq i \leq n$, and $\ell_{i,j} \in \mathrm{Lit}(\Phi)$. We use the same knowledge-base as in the proof of Theorem 3 and use notation from the proof of Theorem 6. However, of course, we will use a more complex program $\mathcal{P}$, which is the union of the following programs

$$\begin{aligned}
&\{\mathsf{Select}_v(0) \leftarrow \neg\mathsf{Select}_v(1);\ \mathsf{Select}_v(1) \leftarrow \neg\mathsf{Select}_v(0) \mid v \in Q\}, \\
&\{\mathsf{Select}_v(0) \vee \mathsf{Select}_v(1) \mid v \in R\}, \\
&\textstyle\bigcup_{i=1}^{n} \{\mathsf{sat}_{C_i}(\theta) \leftarrow \widetilde{\ell}_{i,j} \mid \ell_{i,j} \notin \mathrm{Lit}(S), \theta\colon \bar{S}(C_i) \to \{0,1\}, \\
&\qquad\qquad |\bar{S}(C_i)| < 3, \theta \not\models C_i \cap \mathrm{Lit}(S)\}, \\
&\textstyle\bigcup_{i=1}^{n} \{\mathsf{sat}_{C_i}(\theta) \mid \theta\colon \bar{S}(C_i) \to \{0,1\}, \theta \models C_i\}, \\
&\{\mathsf{sat} \leftarrow \mathsf{sat}_{C_1}(\bar{S}(C_1)), \ldots, \mathsf{sat}_{C_n}(\bar{S}(C_n));\ \leftarrow \neg\mathsf{sat}\}, \\
&\{\mathsf{Select}_v(\alpha) \leftarrow \mathsf{sat} \mid \alpha \in \{0,1\}, v \in R\},
\end{aligned}$$

where sat is a fresh atom.

It remains to argue membership. We can obtain fixed-domain non-entailment of a query $\mathcal{P}$ from a $\mathcal{SROIQ}$ knowledge-base $\mathcal{K}$ ($\mathcal{I} \not\models \mathcal{P}$) by (a) guessing an interpretation $\mathcal{I}$ and verifying whether $\mathcal{I}$ is a model of $\mathcal{K}$ and then (b) by checking whether $\mathcal{P} \cup \mathcal{P}_{\mathcal{I}}$ has a stable model. Since for (a) we can simply guess an interpretation and checking fixed-domain satisfiability of a $\mathcal{SROIQ}$ knowledge-base is in NP (Gaggl, Rudolph, and Schweizer 2016, Lem 5) and for (b) consistency of a bounded-arity Datalog query is in Σ_2^{P} (Eiter et al. 2007, Lem 3), we have an algorithm witnessing membership in $\mathrm{NP}^{\Sigma_3^{\mathrm{P}}} = \Sigma_4^{\mathrm{P}}$. $\square$

Theorem 6. *Let* $\mathcal{C} \in \{\textbf{\textit{Tight}}, \textbf{\textit{Normal}}\}$*. The combined complexity of* $\mathrm{NE}_{\mathcal{C}}$ *w.r.t. a* $\mathcal{SROIQ}$ *or* $\mathsf{DL}_{\mathsf{min}}$ *knowledge-base is* Σ_3^{P}*-complete.*

Characterizing Counting Complexity

In this section, we investigate the complexity of counting (projected) anti-witnesses for entailment problems on bounded-arity Datalog programs. We follow standard terminology in this area (Durand, Hermann, and Kolaitis 2005), using complexity classes preceded with the sharp-dot operator '$\#\cdot$'. A *witness* function is a function $w\colon \Sigma^* \to \mathcal{P}^{<\omega}(\Gamma^*)$, where Σ and Γ are alphabets, mapping to a finite subset of Γ^*. Such a function relates to the counting problem "given $x \in \Sigma^*$, find $|w(x)|$". If $\mathcal{C}$ is a decision complexity class then $\# \cdot \mathcal{C}$ is the class of all counting problems whose witness function w satisfies (1.) $\exists$ polynomial p such that for all $y \in w(x)$, we have that $|y| \leqslant p(|x|)$, and (2.) the decision problem "given x and y, is $y \in w(x)$?" is in $\mathcal{C}$. Finally, note that we use parsimonious reductions (Durand et al. 2005) that preserve the cardinality of solutions between the corresponding witness sets and is computable in polynomial time.

Let $\#\mathrm{NE}_{\mathcal{C}}$; $\mathcal{C} \in \{\textbf{Strat}, \textbf{Tight}, \textbf{Normal}, \textbf{Disj}\}$ be the problem asking to compute the number of anti-witnesses of $\mathcal{K} \models \mathcal{P}$, given A knowledge-base $\mathcal{K}$ and a $\mathcal{C}$-program $\mathcal{P}$. As the reductions in Section 3 preserve the cardinality of solutions, we immediately obtain the following corollary.

Corollary 7. *For a $\mathcal{SROIQ}$ or $\mathsf{DL}_{\mathsf{min}}$ knowledge-base: (1.) $\#\mathrm{NE}_{\textbf{\textit{Strat}}}$ is $\# \cdot$ NP-complete, (2.) $\#\mathrm{NE}_{\textbf{\textit{Tight}}}$ and $\#\mathrm{NE}_{\textbf{\textit{Normal}}}$ are $\# \cdot \Sigma_2^{\mathrm{P}}$-complete, and (3.) $\#\mathrm{NE}_{\textbf{\textit{Disj}}}$ is $\# \cdot \Sigma_3^{\mathrm{P}}$-complete.*

Next, we naturally want to extend the problem above by taking a set of atoms as part of the input to which all solutions are *projected* onto. When we count, multiple anti-witnesses that are identical when restricted to a set of projected atoms those count as only one anti-witness.

Let $\#\mathrm{pNE}_{\mathcal{C}}$; $\mathcal{C} \in \{\textbf{Strat}, \textbf{Tight}, \textbf{Normal}, \textbf{Disj}\}$ be the problem asking to compute the number of distinct anti-witnesses to $\mathcal{K} \models \mathcal{P}$, when restricted to names in Π, given a Datalog program $\mathcal{P} \in \mathcal{C}$, a KB $\mathcal{K}$, and a projection $\Pi \subseteq \mathsf{N}(\mathcal{K})$.

Example 8. *Consider knowledge-base $\mathcal{K}$ and $\mathcal{P}$ from Example 2. Recall that about $\frac{1}{3}$ of the models are anti-witnesses to $\mathcal{K} \models \mathcal{P}$. Still, the number of anti-witnesses can be very large, since one can almost arbitrarily interpret* Fever*. If we are interested only in how to distribute the one mask, we could ask for #pNE on $\mathcal{K}, \mathcal{P}$ and projection $\Pi := \{P_1, \mathsf{Mask}\}$. Then, this results in only three projected anti-witnesses. Each anti-witness $\mathcal{J}$ ensures $P_1^{\mathcal{J}} \in \mathsf{Mask}^{\mathcal{J}}$.* ⊣

Similarly, one can show the result for the projected versions that yield one jump in the complexities, whose proof is in the extended version.

Theorem 9. *For a $\mathcal{SROIQ}$ or $\mathsf{DL}_{\mathsf{min}}$ knowledge-base: (1.) $\#\mathrm{pNE}_{\textbf{\textit{Strat}}}$ is $\# \cdot \Sigma_2^{\mathrm{P}}$-complete, (2.) $\#\mathrm{pNE}_{\textbf{\textit{Tight}}}$ and $\#\mathrm{pNE}_{\textbf{\textit{Normal}}}$ are $\# \cdot \Sigma_3^{\mathrm{P}}$-complete, and (3.) $\#\mathrm{pNE}_{\textbf{\textit{Disj}}}$ is $\# \cdot \Sigma_4^{\mathrm{P}}$-complete.*

4 Algorithms and Results for Treewidth

In the following, we provide a parameterized complexity analysis for the problems NE and #NE when parameterized by treewidth of the input instance. We give algorithms and thereby establish upper bounds on the runtime. These runtime results match with conditional lower bounds if we believe in the exponential time hypothesis (ETH) (Impagliazzo, Paturi, and Zane 2001). Algorithms that exploit small treewidth, typically run *dynamic programming along a tree decomposition* (Bodlaender and Kloks 1996) of a graph representation of the input instance. During dynamic programming the nodes of the decomposition are traversed bottom-up and at each node t of the decomposition a set τ_t of records is computed by means of a bag algorithm. Such a bag algorithm is specified by describing how this set of records is computed for each node of the TD, thereby considering the sets of records for the child node and evaluating only a part of the instance, called bag instance. At the root we interpret the set of records and output the solution to our question.

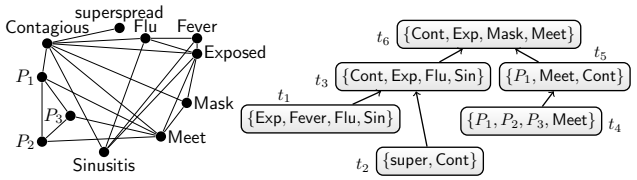

Figure 1: Knowledge-program graph G and a tree-decomposition $\mathcal{T}_G$ of G.

Before we provide our algorithms for treewidth, we require a dedicated graph representation of our knowledge-bases and datalog programs, as well as the definition of a bag instance. Here we are dealing with the non-ground setting and bounding treewidth for both KBs and non-ground datalog programs. Consequently, we define a graph representation similar to the primal graph, but for non-ground datalog and thereby also considering KBs. To this end, we consider for an instance $\mathcal{I} = \langle \mathcal{K}, \mathcal{P} \rangle$, consisting of a given knowledge-base $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ and a given datalog program $\mathcal{P}$, the *knowledge-program graph* $G_{\mathcal{I}}$. The vertices of $G_{\mathcal{I}}$ consist of all individual, concept, and role names of $\mathcal{K}$ as well as all predicates occurring in $\mathcal{P}$. Then, there is an edge between two vertices of $G_{\mathcal{I}}$, whenever the corresponding names appear together in at least one axiom, concept inclusion, role inclusion or at least one datalog rule. Formally, we let $G_{\mathcal{I}} := (V, E)$, where $V := \mathsf{N}(\mathcal{K}) \cup \mathsf{N}_{\mathsf{P}}(\mathcal{P})$ and $E := \{\{e_1, e_2\} \mid \{e_1, e_2\} \subseteq \mathsf{N}(a), a \in \mathcal{A}\} \cup \{\{C, C\prime\} \mid \{C, C\prime\} \subseteq \mathsf{N}(c), c \in \mathcal{T}\} \cup \{\{R, R\prime\} \mid \{R, R\prime\} \subseteq \mathsf{N}(r), r \in \mathcal{R}\} \cup \{\{p_1, p_2\} \mid p_1, p_2 \in \mathsf{N}_{\mathsf{P}}(r), r \in \mathcal{P}\}$.

Example 10. *Consider the knowledge-base $\mathcal{K}$ and program $\mathcal{P}$ from Example 2. The knowledge-program graph G of $\mathcal{K}$ and $\mathcal{P}$ and a decomposition is shown in Figure 1.* ⊣

Let $\mathcal{T} = (T, \chi)$ be a TD of the knowledge-program graph for given $\mathcal{K}$ and $\mathcal{P}$, and let t be a node of T. Then, the *bag knowledge-base* is defined as $\mathcal{K}_t := (\mathcal{A}_t, \mathcal{T}_t, \mathcal{R}_t)$, where $\mathcal{A}_t := \{a \mid a \in \mathcal{A}, \mathsf{N}(a) \subseteq \chi(t)\}$, $\mathcal{T}_t := \{c \mid c \in \mathcal{T}, \mathsf{N}(c) \subseteq \chi(t)\}$, and $\mathcal{R}_t := \{r \mid r \in \mathcal{R}, \mathsf{N}(r) \subseteq \chi(t)\}$. Further, the *bag program* $\mathcal{P}_t$ is given by $\mathcal{P}_t := \{r \mid r \in \mathcal{P}, \mathsf{N}_{\mathsf{P}}(r) \subseteq \chi(t)\}$. As a result, this leads to the *bag instance* $\mathcal{I}_t$, which is defined by $\mathcal{I}_t := \langle \mathcal{K}_t, \mathcal{P}_t \rangle$. Intuitively, the bag instance consists of those axioms and rules, whose symbols are fully contained in the bag.

Example 11. *Consider Example 2 and TD $\mathcal{T}_G$ of G. Take node t_3 from TD $\mathcal{T}_G$. Then, bag knowledge-base $\mathcal{K}_{t_3}$ =*

$(\emptyset, \{\mathsf{Contagious} \equiv \mathsf{Flu} \sqcup \mathsf{Sinusitis}\}, \emptyset)$ *and bag program* $\mathcal{P}_{t_3} = \emptyset$*, resulting in* $\mathcal{I}_{t_3} = \langle \mathcal{K}_{t_3}, \mathcal{P}_{t_3} \rangle$. ⊣

Having established graph representations and the definition of bag instances that is evaluated in a node of a TD of the graph, we are ready to present our bag algorithm for $\mathrm{NE}_{\mathbf{Tight}}$ and $\#\mathrm{NE}_{\mathbf{Tight}}$. While the specifics are for brevity carried out only for the simple case of tight programs, the basic concept of these algorithms vacuously extends to the other program classes, cf., (Fichte and Hecher 2019). This algorithm is described in Figure 2 in the form of a case distinction, whose cases depend on the types of nodes of a nice TD. Note that any TD can be turned into a nice TD without increasing the width, cf., (Kloks 1994). However, for non-nice TDs, these cases simply overlap accordingly. Overall, for any set τ_t of records, Figure 2 maintains records of the following form: $\langle I, C, R, \mathcal{J}, c \rangle$, where $\langle I, C, R \rangle$ forms an interpretation of $\mathcal{K}_t$ and c is an integer for solving #NE (irrelevant for decision problem $\mathrm{NE}_{\mathbf{Tight}}$). Further, $\mathcal{J}$ is a set of tuples of the form $\langle J, P \rangle$, where J is an interpretation of $\mathcal{P}_t$ and $P \subseteq J$ is a set of atoms of J, where for every $a \in P$ there is a rule $r \in \mathcal{P}_t$ *justifying* a. Consequently, we define the atoms of J justified by $\mathcal{P}_t$ by $\mathrm{justif}_t(J) := \{a \mid r \in \mathrm{Ground}(\mathcal{P}_t), a \in H_r, J \subseteq B_r^+, J \cap (B_r^- \cup H_r \setminus \{a\}) = \emptyset\}$.

While the actual details of this bag algorithm are quite technical, the high-level idea of the algorithm for a node t is as follows: The table τ_t for node t contains records consisting of a model of $\mathcal{K}_t$ that can be extended to a model of the knowledge-base obtained by combining all bag knowledge-bases below t in T, as well as a set of models of $\mathcal{P}_t$ such that each of these models can be extended to a stable model of the program consisting of all bag programs below t in T.

For leaf nodes t (see Line 1 of Figure 2) the set τ_t of records only contains the empty interpretation (satisfying the empty knowledge-base) and the empty interpretation and the empty set of justified atoms satisfying the empty program.

Now, let $s \in \mathsf{N}(\mathcal{K}_t) \cup \mathsf{N}_\mathsf{P}(\mathcal{P}_t)$ be any symbol of the bag instance $\mathcal{I}_t$. Then, we define the (exhaustive) interpretations for individuals by $\mathrm{In}_t(s) := \{s \mapsto d \mid s \in \mathsf{N}_\mathsf{I}(\mathcal{K}_t), d \in \Delta\}$, the concept interpretations by $\mathrm{Co}_t(s) := \{s \mapsto D \mid s \in \mathsf{N}_\mathsf{C}(\mathcal{K}_t), D \subseteq \Delta\}$, and the role interpretations by $\mathrm{Ro}_t(s) := \{s \mapsto \mathcal{D} \mid s \in \mathsf{N}_\mathsf{R}(\mathcal{K}_t), \mathcal{D} \subseteq (\Delta \times \Delta)\}$, as well as the datalog interpretations by $\mathrm{Pr}_t(s) := \{s(d) \mid s \in \mathsf{N}_\mathsf{P}(\mathcal{P}_t), d \in \Delta\}$. These definitions are used to exhaustively guess for nodes t that introduce a symbol s, as shown in Line 3 of Figure 2, all potential interpretations satisfying both $\mathcal{K}_t$, as well as $\mathcal{P}_t$. Further, Line 3 also "updates" justified atoms.

Intuitively, whenever a symbol s is forgotten in a node t (cf., Line 5), the interpretations do not need to care about s any more, so we remove information about s. However, we have to ensure that every atom in $\mathcal{J}$ is justified. Further, for forget nodes one has to take care that interpretations that are different for child nodes of t might collapse in the set of records τ_t, so one has to sum up the corresponding counters.

For a join node t, which is a node with two child nodes as handled by Line 8, one has to ensure to combine records, where the corresponding assignments coincide. Justifications for atoms, however, suffice in only one node and one has to multiply counters accordingly.

Finally, we analyze the set τ_r of records for the (empty) root node r of TD $\mathcal{T}$. Instance $\mathcal{I}$ is a positive instance of $\mathrm{NE}_{\mathbf{Tight}}$ if and only if τ_r contains a record u whose fourth component $\mathcal{J}$ is empty. It is easy to see that also the complement of $\mathrm{NE}_{\mathbf{Tight}}$ can be solved using Figure 2, i.e., therefore (runtime) properties sustain even for the complement problem. Further, the number of solutions to problem $\#\mathrm{NE}_{\mathbf{Tight}}$ is contained in the fifth component (counter c) of this record u.

In: Node t, bag $\chi(t)$, bag instance $\mathcal{I}_t = (\mathcal{K}_t, \mathcal{P}_t)$, and sets $\langle \tau_1, \ldots, \tau_\ell \rangle$ of records for children of t.
Out: Set τ_t of records.

1 **if** $\mathrm{type}(t) =$ *leaf* **then** $\tau_t \leftarrow \{\langle \emptyset, \emptyset, \emptyset, \{\langle \emptyset, \emptyset \rangle\}, 1 \rangle\}$
2 **else if** $\mathrm{type}(t) =$ *intr* **and** *symbol* $s \in \chi(t)$ *is introduced* **then**
3 $\tau_t \leftarrow \{\langle I', C', R', \mathcal{J}', c \rangle \mid \langle I, C, R, \mathcal{J}, c \rangle \in \tau_1, I \subseteq I' \subseteq \mathrm{In}_t(s)^+_I, C \subseteq C' \subseteq \mathrm{Co}_t(s)^+_C, R \subseteq R' \subseteq \mathrm{Ro}_t(s)^+_R, \mathcal{J}' = \{\langle J', \mathrm{justif}_t(X)^+_P \rangle \mid \langle J, P \rangle \in \mathcal{J}, J \subseteq J' \subseteq \mathrm{Pr}_t(s)^+_J, X = I' \cup C' \cup R' \cup J', X \models \mathcal{P}_t\}, \langle I', C', R' \rangle \models \mathcal{K}_t\}$
4 **else if** $\mathrm{type}(t) =$ *forget* **and** *symbol* $s \notin \chi(t)$ *is forgotten* **then**
5 $\rho_t \leftarrow \{\langle I^-_{\mathrm{In}_t(s)}, C^-_{\mathrm{Co}_t(s)}, R^-_{\mathrm{Ro}_t(s)}, \{\langle J^-_{\mathrm{Pr}_t(s)}, P^-_{\mathrm{Pr}_t(s)} \rangle \mid \langle J, P \rangle \in \mathcal{J}, J \cap \mathrm{Pr}_t(s) \subseteq P\}, c, u \rangle \mid u \in \tau_1, u = \langle I, C, R, \mathcal{J}, c \rangle\}$
6 $\tau_t \leftarrow \{\langle I', C', R', \mathcal{J}', \sum_{\langle I', C', R', \mathcal{J}', c, \ldots \rangle \in \rho_t} \{c\} \rangle \mid \langle I', C', R', \mathcal{J}', \ldots \rangle \in \rho_t\}$
7 **else if** $\mathrm{type}(t) =$ *join* **then**
8 $\tau_t \leftarrow \{\langle I, C, R, \{\langle J, P_1 \cup P_2 \rangle \mid \langle J, P_1 \rangle \in \mathcal{J}_1, \langle J, P_2 \rangle \in \mathcal{J}_2\}, c_1 \cdot c_2 \rangle \mid \langle I, C, R, \mathcal{J}_1, c_1 \rangle \in \tau_1, \langle I, C, R, \mathcal{J}_2, c_2 \rangle \in \tau_2\}$

$S^+_{S'} := S \cup S'$ and $S^-_{S'} := S \setminus S'$.

Figure 2: Bag algorithm $\#\mathrm{NE}_{\mathbf{Tight}}(t, \langle \tau_1, \ldots, \tau_\ell \rangle)$.

Example 12. *Consider the knowledge-base* $\mathcal{K}$*, domain* Δ *and program* $\mathcal{P}$ *from Example 2 as well as TD* $\mathcal{T}$ *of* $G_{\langle \mathcal{K}, \mathcal{P} \rangle}$*. Then, table* τ_4 *for node* t_4 *is the result of introducing symbols (individual names)* P_1, P_2, P_3 *as well as role name* Meet*. Consequently, due to Line 3 of Figure 2, we have* $\langle \{P_1 \mapsto \mathrm{donald}, P_2 \mapsto \mathrm{emanuel}, P_3 \mapsto \mathrm{silvio}\}, \emptyset, \{\mathsf{Meet} \mapsto \{(\mathrm{donald}, \mathrm{emanuel}), (\mathrm{emanuel}, \mathrm{silvio})\}\}, \{\langle \emptyset, \emptyset \rangle\}, 1 \rangle \in \tau_4$*. In total there are* $3! = 3 \cdot 2 \cdot 1 = 6$ *many records of similar kind in* τ_4*, corresponding to the possibilities of mapping individuals* P_1, P_2, P_3 *to* Δ *without overlaps. For node* t_2*, we introduce* superspread *as well as* Contagious*. Note that* superspread $\notin \chi(t_3)$ *and therefore only those records of table* τ_2 *have successor records in table* τ_3 *for node* t_3*, where we have justification for the atom* superspread *of arity* 0 *over predicate* superspread*, cf., Line 5 of Figure 2. Among these sustaining records we have* $\langle \emptyset, \{\mathsf{Contagious} \mapsto \{\mathrm{donald}, \mathrm{emanuel}, \mathrm{silvio}\}\}, \emptyset, \{\langle \{\mathsf{superspread}\}, \{\mathsf{superspread}\} \rangle\}, 1 \rangle$ *of* τ_2 *as well as* $\binom{3}{2} = 3$ *many records according to the possibilities of selecting two contagious people over domain* Δ *in order to be able to derive and justify* superspread *as required by program* $\mathcal{P}$*. Then, node* t_3 *is the result of removing from table* τ_2 *predicate name* superspread *and merging the result (cf., Line 8) with the table obtained from table* τ_1 *for node* t_1*, where interpretations for concept name* Fever *have been removed, cf., Line 5. Similarly, one continues with computing the table for node* t_6 *in order to finally obtain that there are anti-witnesses of* $\mathcal{K} \not\models \mathcal{P}$ *and the solution to* $\mathrm{NE}_{\mathbf{Tight}}$*.* ⊣

Overall, the algorithm established in Figure 2 leads to the following result due to the number of records and since a TD of $G_{\mathcal{K}, \mathcal{P}}$ with a linear number of nodes can be 5-

approximated (Bodlaender et al. 2016) in linear time.

Theorem 13. *Given a KB $\mathcal{K}$ and a program $\mathcal{P} \in$ **Tight**, both over fixed domain Δ, where k is the treewidth of the knowledge-program graph $G_{\langle \mathcal{K},\mathcal{P}\rangle}$. Then, we can decide $\mathrm{NE}_{\textbf{\textit{Tight}}}$ in time $\mathsf{tower}(2, |\Delta|^{\mathcal{O}(k)}) \cdot |\mathsf{N}(\mathcal{K}) \cup \mathsf{N}_\mathsf{P}(\mathcal{P})|$.*

This approach can be used for the class **Strat** of stratified programs, where the fourth components $\mathcal{J}$ of the records only need to be of cardinality 1, since stratified programs have one unique stable model (Apt, Blair, and Walker 1988). Further, the algorithm can be extended to more expressive datalog programs, namely normal and disjunctive programs, thereby obtaining runtimes of $\mathsf{tower}(2, |\Delta|^{\mathcal{O}(k\cdot\log(k))}) \cdot |\mathsf{N}(\mathcal{K}) \cup \mathsf{N}_\mathsf{P}(\mathcal{P})|$ and $\mathsf{tower}(3, |\Delta|^{\mathcal{O}(k)}) \cdot |\mathsf{N}(\mathcal{K}) \cup \mathsf{N}_\mathsf{P}(\mathcal{P})|$, respectively. For counting, we need to keep track of counts, which slightly increases runtimes from being linear in the instance size $|\mathsf{N}(\mathcal{K}) \cup \mathsf{N}_\mathsf{P}(\mathcal{P})|$ to a runtime that is subquadratic in the instance size. The reason is that the multiplication of two n-bit numbers, if not assumed to be constant-time by definition, runs in time $n \cdot \log(n) \cdot \log(\log(n))$ (Knuth 1998).

Similarly, one can provide dynamic programming algorithms for problem $\#\mathrm{pNE}_\mathcal{C}$ by lifting algorithms for projected counting of the literature (Fichte and Hecher 2019). Thereby, the runtime results are exponentially worsened in the treewidth, cf., complexity results summarized in Table 2.

Hardness Results via Conditional Lower Bounds

However, if we believe in reasonable assumptions in computational complexity, namely the *exponential time hypothesis (ETH)* (Impagliazzo, Paturi, and Zane 2001), it turns out that one can show that these runtimes can probably not be significantly improved in the worst case. ETH is a widely accepted standard hypothesis in algorithms and complexity, which implies that one cannot solve the Boolean satisfiability problem in sub-exponential time.

To this end, we rely on very recent conditional lower bounds (Fichte, Hecher, and Kieler 2020, Thm. 15) for quantified constraint satisfaction and show the following.

Theorem 14. *Given a knowledge-base $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ and a datalog program $\mathcal{P}$, both over fixed domain Δ, such that the treewidth of the knowledge-program graph $G_{\mathcal{K},\mathcal{P}}$ is k and assume ETH. Then, problems $\mathrm{NE}_\mathcal{C}, \#\mathrm{NE}_\mathcal{C}$ for $\mathcal{C} =$ **Strat** and $\mathcal{C} \in \{$**Tight**, **Normal**$\}$ can not be solved in time $f(k) \cdot \mathrm{poly}(|\mathsf{N}(\mathcal{K}) \cup \mathsf{N}_\mathsf{P}(\mathcal{P})|)$ with $f(k) = 2^{|\Delta|^{o(k)}}$ and $f(k) = \mathsf{tower}(2, |\Delta|^{o(k)})$, respectively. Further, problems $\mathrm{NE}_{\textbf{\textit{Disj}}}, \#\mathrm{NE}_{\textbf{\textit{Disj}}}$ can not be solved in time $\mathsf{tower}(3, |\Delta|^{o(k)}) \cdot \mathrm{poly}(|\mathsf{N}(\mathcal{K}) \cup \mathsf{N}_\mathsf{P}(\mathcal{P})|)$.*

Proof (Sketch). We show the case for $\mathcal{C} =$ **Disj** by reducing $\mathsf{QCSPVAL}_{\forall,4}$ to the complement of $\mathrm{NE}_{\textbf{Disj}}$. Let $k, m, h, o \in \mathbb{N}$, $P := \{p_1, \ldots, p_k\}$, $Q := \{q_1, \ldots, q_m\}$, $R := \{r_1, \ldots, r_h\}$, and $S := \{s_1, \ldots, s_o\}$. Further, let $\Phi = \forall p_1, \ldots, p_k \exists q_1, \ldots, q_m \forall r_1, \ldots, r_h \exists s_1, \ldots, s_o \mathcal{C}$, where each constraint C_i in $\mathcal{C} = \{C_1, \ldots, C_n\}$ is over three variables $v_{i,1}, \ldots, v_{i,l_i} \in P \cup Q \cup R$ and $v_{i,l_i+1}, \ldots v_{i,3} \in S$, and forms allowed assignments $\{\sigma_1, \ldots, \sigma_{|C_i|}\}$ over Δ.

Now, we will define a DL_min knowledge-base $\mathcal{K} = (\mathcal{A}, \emptyset, \emptyset)$ over domain Δ, and a non-ground Datalog program $\mathcal{P}$ such that $\mathcal{K} \models \mathcal{P}$ if and only if Φ is valid.

	Strat	**Tight**	**Normal**	**Disj**
$\mathrm{NE}_\mathcal{C}$	Σ_2^P	Σ_3^P	Σ_3^P	Σ_4^P
$(\#)\mathrm{NE}_\mathcal{C}[tw]$	$2^{\lvert\Delta\rvert^{\Theta(k)}}$	$2^{2^{\lvert\Delta\rvert^{\Theta(k)}}}$	$2^{2^{\lvert\Delta\rvert^{\mathcal{O}(k\cdot\log(k))}}}$	$2^{2^{2^{\lvert\Delta\rvert^{\Theta(k)}}}}$
$\#\mathrm{NE}_\mathcal{C}$	$\# \cdot \mathrm{NP}$	$\# \cdot \Sigma_2^\mathrm{P}$	$\# \cdot \Sigma_2^\mathrm{P}$	$\# \cdot \Sigma_3^\mathrm{P}$
$\#\mathrm{pNE}_\mathcal{C}$	$\# \cdot \Sigma_2^\mathrm{P}$	$\# \cdot \Sigma_3^\mathrm{P}$	$\# \cdot \Sigma_3^\mathrm{P}$	$\# \cdot \Sigma_4^\mathrm{P}$
$\#\mathrm{pNE}_\mathcal{C}[tw]$	$2^{2^{\lvert\Delta\rvert^{\Theta(k)}}}$	$2^{2^{2^{\lvert\Delta\rvert^{\Theta(k)}}}}$	$2^{2^{2^{\lvert\Delta\rvert^{\mathcal{O}(k\cdot\log(k))}}}}$	$2^{2^{2^{2^{\lvert\Delta\rvert^{\Theta(k)}}}}}$

Table 2: Combined complexity overview of the non-entailment problem variants over fixed domain Δ on specific bounded-arity Datalog programs from class $\mathcal{C}$ w.r.t. a $\mathcal{SROIQ}$ or DL_min knowledge-bases. For the tw-result, we assume ETH and omit polynomial factors in the instance size. All entries are completeness results or provide matching and lower bounds (under ETH), except parameterized results for class **Normal**, where we only have the slightly weaker lower bounds as for **Tight**.

Further, let the ABox $\mathcal{A}$ contain the following axioms: $\mathsf{Select}_v(x_v)$ for all $v \in \{p_1, \ldots, p_k\}$. We define program $\mathcal{P}$ as the union of following programs $\{ \bigvee_{d\in\Delta} \mathsf{Select}_v(d) \mid v \in Q \}$, $\{ \bigvee_{d\in\Delta} \mathsf{Select}_v(d) \mid v \in R \}$, $\bigcup_{i=1}^n \{ \mathsf{sat}_{C_i}(\theta) \leftarrow \mathsf{Select}_{v_{i,1}}(\sigma(v_{i,1})), \ldots, \mathsf{Select}_{v_{i,l_i}}(\sigma(v_{i,l_i})) \mid \theta\colon \bar{S}(C_i) \to \Delta, \sigma \in C_i, l_i \geq 1, \theta = \sigma|_{\bar{S}(C_i)} \}$, $\bigcup_{i=1}^n \{ \mathsf{sat}_{C_i}(\theta) \mid \theta\colon \bar{S}(C_i) \to \Delta, \sigma \in C_i, l_i = 0, \theta = \sigma|_{\bar{S}(C_i)} \}$, $\{\mathsf{sat} \leftarrow \mathsf{sat}_{C_1}(\bar{S}(C_1)), \ldots, \mathsf{sat}_{C_n}(\bar{S}(C_n)); \leftarrow \neg\mathsf{sat}\}$, $\{\mathsf{Select}_v(d) \leftarrow \mathsf{sat} \mid v \in R, d \in \Delta\}$,where $\sigma_{|X}$ is function σ, but restricted to variables in X. Then, Φ is valid if and only if the entailment holds (also for every subset-minimal model). Observe that this also shows hardness for $\#\mathrm{NE}_{\textbf{Disj}}$. Further, compared to the treewidth k of instance Σ, the treewidth k' of $G_{\mathcal{K},\mathcal{P}}$ is linear in k. The only large rule $r \in \mathcal{P}$ is the one with $\mathsf{sat} \in H_r$, which can be split up in auxiliary rules such that k' is linear in k. □

The same idea can be used to lift these lower bounds (cf., Theorem 9) for the problem #pNE, as provided in Table 2.

5 Conclusion

We provide a wide view on the combined complexity of the fixed-domain non-entailment problem NE on specific bounded-arity Datalog programs with respect to a $\mathcal{SROIQ}$ or DL_min knowledge-bases. This approach combines open and closed world semantics in a very general way. We investigate the classical decision and counting complexity as well as the parameterized complexity of this problem when allowing for arbitrary disjunctive Datalog programs or restricting the program to be stratified, normal, or tight. Interestingly, by applying very recent advances in parameterized complexity, we immodestly obtain tight complexity results.

A further research direction is to study our algorithm for practical applications and stronger parameters, e.g., (Grohe and Marx 2014; Greco et al. 2018), particularly, in the light of recent decomposition techniques (Gottlob, Okulmus, and Pichler 2020). Also enumeration complexity (Creignou et al. 2017, 2019; Meier 2020) is worth studying in this context.

Acknowledgements

The work has been supported by FWF, Grants Y698, and P32830, WWTF Grant ICT19-065, and the German Research Foundation (DFG) under the project number ME 4279/1-2. Markus Hecher is also affiliated with the University of Potsdam, Germany. The authors would like to thank the anonymous reviewers for the valuable feedback they have provided.

References

Apt, K. R.; Blair, H. A.; and Walker, A. 1988. Towards a theory of declarative knowledge. *Foundations of deductive databases and logic programming* 89–148.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press, Cambridge. ISBN 9781139025355.

Baader, F.; Küsters, R.; and Wolter, F. 2003. Extensions to Description Logics. In *Description Logic Handbook*, 219–261. Cambridge University Press.

Baader, F.; Lutz, C.; Milicic, M.; Sattler, U.; and Wolter, F. 2005. Integrating Description Logics and Action Formalisms: First Results. In *AAAI'05*, 572–577.

Barceló, P.; Feier, C.; Lutz, C.; and Pieris, A. 2019. When is Ontology-Mediated Querying Efficient? In *LICS'19*, 1–13.

Bienvenu, M.; Kikot, S.; Kontchakov, R.; Ryzhikov, V.; and Zakharyaschev, M. 2017. Optimal Nonrecursive Datalog Rewritings of Linear TGDs and Bounded (Hyper)Tree-Width Queries. In Artale, A.; Glimm, B.; and Kontchakov, R., eds., In *DL'17*.

Bienvenu, M.; Ortiz, M.; Šimkus, M.; and Xiao, G. 2013. Tractable Queries for Lightweight Description Logics. In *IJCAI'13*, 768–774. AAAI Press. ISBN 9781577356332.

Bodlaender, H. L.; Drange, P. G.; Dregi, M. S.; Fomin, F. V.; Lokshtanov, D.; and Pilipczuk, M. 2016. A c^k n 5-Approximation Algorithm for Treewidth. *SIAM J. Comput.* 45(2): 317–378.

Bodlaender, H. L.; and Kloks, T. 1996. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms* 21(2): 358–402.

Calvanese, D. 2006. Finite Model Reasoning in Description Logics. In Patrick Doherty, John Mylopoulos, C. A. W., ed., *KR'06*, 292–303.

Creignou, N.; Ktari, R.; Meier, A.; Müller, J.; Olive, F.; and Vollmer, H. 2019. Parameterised Enumeration for Modification Problems. *Algorithms* 12(9): 189.

Creignou, N.; Meier, A.; Müller, J.; Schmidt, J.; and Vollmer, H. 2017. Paradigms for Parameterized Enumeration. *Theory Comput. Syst.* 60(4): 737–758.

de Haan, R. 2018. Hunting for Tractable Languages for Judgment Aggregation. In Thielscher, M.; Toni, F.; and Wolter, F., eds., *KR'18*, 194–203. AAAI Press.

Dudek, J. M.; Phan, V. H. N.; and Vardi, M. Y. 2020. ADDMC: Weighted Model Counting with Algebraic Decision Diagrams. In Conitzer, V.; and Sha, F., eds., *AAAI'20*, 1468–1476.

Durand, A.; Hermann, M.; and Kolaitis, P. G. 2005. Subtractive reductions and complete problems for counting complexity classes. *Theoretical Computer Science* 340(3): 496–513. ISSN 0304–3975.

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

Eiter, T.; and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell.* 15(3–4): 289–323.

Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence* 172(12): 1495–1539. ISSN 0004-3702.

Eiter, T.; Pan, J. Z.; Schneider, P.; Šimkus, M.; and Xiao, G. 2015. A Rule-based Framework for Creating Instance Data from OpenStreetMap. In ten Cate, B.; and Mileo, A., eds., *RR'15*, 93–104. Springer. ISBN 978-3-319-22002-4.

Fichte, J. K.; and Hecher, M. 2019. Treewidth and Counting Projected Answer Sets. In *LPNMR'19*, volume 11481 of *LNCS*, 105–119. Springer.

Fichte, J. K.; Hecher, M.; and Hamiti, F. 2020. The Model Counting Competition 2020. *CoRR* 2012.01323.

Fichte, J. K.; Hecher, M.; and Kieler, M. F. I. 2020. Treewidth-Aware Quantifier Elimination and Expansion for QCSP. In *CP'20*. Springer.

Fichte, J. K.; Hecher, M.; and Meier, A. 2018. Counting Complexity for Reasoning in Abstract Argumentation. In Hentenryck, P. V.; and Zhou, Z.-H., eds., *AAAI'19*.

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2017. Answer Set Solving with Bounded Treewidth Revisited. In Balduccini, M.; and Janhunen, T., eds., *LPNMR'17*, volume 10377 of *LNCS*, 132–145. Springer. ISBN 978-3-319-61660-5.

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2018a. Exploiting Treewidth for Projected Model Counting and its Limits. In *SAT'18*, volume 10929 of *LNCS*, 165–184. Springer. ISBN 978-3-319-94144-8.

Fichte, J. K.; Hecher, M.; and Pfandler, A. 2020. Lower Bounds for QBFs of Bounded Treewidth. In Kobayashi, N., ed., *LICS'20*, 410–424. Assoc. Comput. Mach., New York.

Fichte, J. K.; Hecher, M.; and Schindler, I. 2018. Default Logic and Bounded Treewidth. In Klein, S. T.; and Martín-Vide, C., eds., *LATA'18*, LNCS. Springer.

Fichte, J. K.; Hecher, M.; Thier, P.; and Woltran, S. 2020. Exploiting Database Management Systems and Treewidth for Counting. In *PADL*, volume 12007 of *LNCS*, 151–167. Springer.

Fichte, J. K.; Hecher, M.; Woltran, S.; and Zisser, M. 2018b. Weighted Model Counting on the GPU by Exploiting Small Treewidth. In Azar, Y.; Bast, H.; and Herman, G.,

eds., *ESA'18*, volume 112, 28:1–28:16. Dagstuhl Publishing. ISBN 978-3-95977-081-1.

Gaggl, S. A.; Rudolph, S.; and Schweizer, L. 2016. Fixed-Domain Reasoning for Description Logics. In Kaminka, G. A.; Fox, M.; Bouquet, P.; Hüllermeier, E.; Dignum, V.; Dignum, F.; and van Harmelen, F., eds., *ECAI'16*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 819–827.

Gelfond, M.; and Lifschitz, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Comput.* 9(3/4): 365–386.

Gottlob, G.; Okulmus, C.; and Pichler, R. 2020. Fast and Parallel Decomposition of Constraint Satisfaction Problems. In Bessiere, C., ed., *IJCAI'20*, 1155–1162.

Gottlob, G.; Pichler, R.; and Wei, F. 2006. Bounded Treewidth as a Key to Tractability of Knowledge Representation and Reasoning. In *AAAI'06*, 250–256.

Gottlob, G.; Pichler, R.; and Wei, F. 2007. Efficient datalog abduction through bounded treewidth. In *AAAI*, 1626–1631.

Gottlob, G.; and Szeider, S. 2007. Fixed-Parameter Algorithms For Artificial Intelligence, Constraint Satisfaction and Database Problems. *The Computer Journal* 51(3): 303–325. ISSN 0010-4620.

Greco, G.; Leone, N.; Scarcello, F.; and Terracina, G. 2018. Structural Decomposition Methods: Key Notions and Database Applications. In Flesca, S.; Greco, S.; Masciari, E.; and Saccà, D., eds., *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, 253–267. Springer Verlag. ISBN 978-3-319-61893-7.

Grohe, M. 2007. The Complexity of Homomorphism and Constraint Satisfaction Problems Seen from the Other Side. *J. ACM* 54(1). ISSN 0004-5411.

Grohe, M.; and Marx, D. 2014. Constraint solving via fractional edge covers. *ACM Transactions on Algorithms* 11(1): Art. 4, 20.

Hecher, M.; Thier, P.; and Woltran, S. 2020. Taming High Treewidth with Abstraction, Nested Dynamic Programming, and Database Technology. In *Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing (SAT'20)*, 343–360. Springer Verlag.

Hitzler, P.; Parsia, K. B.; Patel-Schneider, P. F.; and Rudolph, S. 2012. OWL 2 Web Ontology Language Primer. Technical report, W3C.

Horrocks, I.; Kutz, O.; and Sattler, U. 2006. The Even More Irresistible SROIQ. In Patrick Doherty, John Mylopoulos, C. A. W., ed., *KR'06*,57–67. AAAI Press. ISBN 9781577352716.

Horrocks, I.; and Sattler, U. 2007. A Tableau Decision Procedure for $\mathcal{SHOIQ}$. *J. Autom. Reason.* 39(3): 249–276. ISSN 0168-7433.

Huang, S.; Li, Q.; and Hitzler, P. 2012. Reasoning with inconsistencies in hybrid MKNF knowledge bases. *Logic Journal of the IGPL* 21(2): 263–290. ISSN 1367-0751.

Impagliazzo, R.; Paturi, R.; and Zane, F. 2001. Which Problems Have Strongly Exponential Complexity? *J. of Computer and System Sciences* 63(4): 512–530. ISSN 0022-0000.

Klinov, P. 2008. Pronto: A Non-monotonic Probabilistic Description Logic Reasoner. In Bechhofer, S.; Hauswirth, M.; Hoffmann, J.; and Koubarakis, M., eds., *ESWC'08*, 822–826. Springer. ISBN 978-3-540-68234-9.

Kloks, T. 1994. *Treewidth. Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer Verlag. ISBN 3-540-58356-4.

Knublauch, H.; Musen, M. A.; and Rector, A. L. 2004. Editing Description Logic Ontologies with the Protégé OWL Plugin. In Haarslev, V.; and Möller, R., eds., *DL'04*, volume 104.

Knuth, D. E. 1998. How fast can we multiply? In *The Art of Computer Programming*, volume 2 of *Seminumerical Algorithms*, chapter 4.3.3, 294–318. Addison-Wesley, 3 edition.

Krötzsch, M.; Simancik, F.; and Horrocks, I. 2012. A Description Logic Primer. *CoRR* abs/1201.4089.

Lin, F.; and Zhao, J. 2003. On tight logic programs and yet another translation from normal logic programs to propositional logic. In Gottlob, G.; and Walsh, T., eds., *IJCAI'03*, 853–858. Morgan Kaufmann.

Ma, Y.; and Hitzler, P. 2010. Distance-based Measures of Inconsistency and Incoherency for Description Logics. In Haarslev, V.; Toman, D.; and Weddell, G. E., eds., *DL'10*, volume 573.

Ma, Y.; Qi, G.; and Hitzler, P. 2011. Computing inconsistency measure based on paraconsistent semantics. *J. Logic Comput.* 21(6): 1257–1281. ISSN 0955-792X.

Meier, A. 2020. *Parametrised enumeration*. Habilitation thesis, Leibniz Universität Hannover.

Motik, B. 2006. *Reasoning in description logics using resolution and deductive databases*. Ph.D. thesis, Karlsruhe Institute of Technology, Germany.

Pan, G.; and Vardi, M. Y. 2006. Fixed-Parameter Hierarchies inside PSPACE. In *LICS'06*, 27–36. IEEE Computer Society.

Rosati, R. 2011. On the Complexity of Dealing with Inconsistency in Description Logic Ontologies. In Walsh, T., ed., *IJCAI'11*, 1057–1062.

Simančík, F.; Motik, B.; and Horrocks, I. 2014. Consequence-Based and Fixed-Parameter Tractable Reasoning in Description Logics. *Artificial Intelligence* 209: 29–77. ISSN 0004-3702.