Justifications for Description Logic Knowledge Bases Under the Fixed-Domain Semantics

Sebastian Rudolph[1], Lukas Schweizer[1(✉)], and Satyadharma Tirtarasa[2]

[1] Institute of Artificial Intelligence, TU Dresden, Dresden, Germany
{sebastian.rudolph,lukas.schweizer}@tu-dresden.de
[2] Institute of Theoretical Computer Science, TU Dresden, Dresden, Germany
satyadharma.tirtarasa@tu-dresden.de

Abstract. The fixed-domain semantics for OWL and description logic has been introduced to open up the OWL modeling and reasoning tool landscape for use cases resembling constraint satisfaction problems. While standard reasoning under this new semantics is by now rather well-understood theoretically and supported practically, more elaborate tasks like computation of justifications have not been considered so far, although being highly important in the modeling phase. In this paper, we compare three approaches to this problem: one using standard OWL technology employing an axiomatization of the fixed-domain semantics, one using our dedicated fixed-domain reasoner `Wolpertinger` in combination with standard justification computation technology, and one where the problem is encoded entirely into answer-set programming.

1 Introduction

With the success of semantic technologies and its tool support, most notably the OWL language family and its status as W3C standard, more and more people from various application domains create and use ontologies. Meanwhile, ontological modeling is not only well supported by established tools like Protégé, also methodologies such as the usage of ontology design patterns help practitioners to design and deploy ontologies of high quality [9].

Despite these evolutionary improvements in ontology engineering, the resulting ontologies are not free of errors such as unintended entailments (including the case of inconsistency). For that purpose, research has already brought up several techniques to detect the causalities of unintended entailments, and it has been studied for lightweight ontology languages such as $\mathcal{EL}$ [22], as well as for very expressive description logics up to $\mathcal{SROIQ}$ [13,15], which in fact is the logical foundation of OWL 2 DL [10]. These techniques already found their way as built-in functionality into tools like Protégé, or are available stand-alone. In any case, these methods have become an integral part of the semantic development chain.

C. Benzmüller et al. (Eds.): RuleML+RR 2018, LNCS 11092, pp. 185–200, 2018.
https://doi.org/10.1007/978-3-319-99906-7_12

When considering their purpose, ontologies are often divided into two groups: those where the intended use is an (highly axiomatized) expert system focusing on automated reasoning as main use (typically less data driven), or those ontologies that are rather used for data sharing, integration, and reuse with little or no intentions on reasoning (typically data driven) [9]. However in our collaborations with practitioners, we found scenarios exhibiting characteristics of both usages, aiming at ontologies that (a) represent a detailed specification of some product (schema knowledge), (b) include all data and (c) contain axioms that (non-deterministically) specify variable (configurable) parts of the product. In general, these ontologies resemble constraint-type problems, where the purpose of typical automated reasoning tasks is (i) checking satisfiability and (ii) asking for models – solutions of the encoded problem. For both tasks, the natural assumption in this setup is that the domain is explicitly given in the ontology, and thus is finite and fixed a priori.

To accommodate these requirements, the *fixed-domain semantics* has been introduced [6,27], which allows for reasoning over an explicitly given finite domain. A reasoner, named `Wolpertinger`[1], that supports standard reasoning as well as model enumeration under the fixed-domain semantics has been developed [28], based on a translation of DL into answer-set programming.

Our motivation in this paper is to elaborate on possible approaches to compute justifications for ontologies under the fixed-domain semantics. We focus on three approaches that evolved naturally during our investigation. First, it is possible to axiomatize a finite domain and conduct fixed-domain reasoning using standard tools, such that computing explanations can be done via standard tools as well. Second, the `Wolpertinger` reasoner can be coupled with the off-the-shelf justification components of Protégé, and finally we introduce a dedicated encoding of the whole problem into answer-set programming. Our contributions in this paper are:

1. A formal framework for justifications under the fixed-domain semantics.
2. A novel translation for $\mathcal{SROIQ}$ into answer-set programming that allows for standard reasoning and model enumeration.
3. An extended version of the translation enabling to compute justifications where the problem is encoded entirely into answer-set programming.
4. A comparison of three different approaches: one using standard OWL technology employing an axiomatization of the fixed-domain semantics, one using our dedicated fixed-domain reasoner `Wolpertinger` in combination with standard justification computation technology, and one with our novel translation where the problem is encoded entirely into answer-set programming.

The paper is organized as follows. We briefly recall the description logic $\mathcal{SROIQ}$ and a sufficient background on answer-set programming in Sect. 2. In Sect. 3, we introduce the notion of justifications, especially under the fixed-domain semantics. Each possible approach to compute justifications is then depicted in detail in Sect. 4. Finally, we compare the introduced methodologies in Sect. 5.

[1] https://github.com/wolpertinger-reasoner.

2 Preliminaries

OWL 2 DL, the version of the Web Ontology Language we focus on, is defined based on description logics (DLs, [1,26]). We briefly recap the description logic $\mathcal{SROIQ}$ (for details see [14]). Let N_I, N_C, and N_R be finite, disjoint sets called *individual names*, *concept names* and *role names* respectively. These atomic entities can be used to form complex ones as displayed in Table 1.

A $\mathcal{SROIQ}$ *knowledge base* $\mathcal{K}$ is a tuple $(\mathcal{A}, \mathcal{T}, \mathcal{R})$ where $\mathcal{A}$ is an ABox, $\mathcal{T}$ is a TBox and $\mathcal{R}$ is an RBox. Table 2 presents the respective axiom types available in the three parts. The definition of $\mathcal{SROIQ}$ also contains so-called *global restrictions* which prevents certain axioms from occurring together, retaining decidability. They are not necessary for fixed-domain reasoning considered here, hence we omit them for the sake of brevity. We use $N_I(\mathcal{K})$, $N_C(\mathcal{K})$, and $N_R(\mathcal{K})$ to denote the sets of individual names, concept names, and role names occurring in $\mathcal{K}$, respectively.

The semantics of $\mathcal{SROIQ}$ is defined via interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ composed of a non-empty set $\Delta^\mathcal{I}$ called the *domain of* $\mathcal{I}$ and a function $\cdot^\mathcal{I}$ mapping individual names to elements of $\Delta^\mathcal{I}$, concept names to subsets of $\Delta^\mathcal{I}$, and role names to subsets of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$. This mapping is extended to complex role and concept expressions (cf. Table 1) and finally used to define satisfaction of axioms (see Table 2). We say that $\mathcal{I}$ *satisfies* a knowledge base $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ (or $\mathcal{I}$ is a *model* of $\mathcal{K}$, written: $\mathcal{I} \models \mathcal{K}$) if it satisfies all axioms of $\mathcal{A}$, $\mathcal{T}$, and $\mathcal{R}$. We say that a knowledge base $\mathcal{K}$ *entails* an axiom α (written $\mathcal{K} \models \alpha$) if all models of $\mathcal{K}$ are models of α.

Table 1. Syntax and semantics of role and concept constructors in $\mathcal{SROIQ}$, where $a_1, \ldots a_n$ denote individual names, s a role name, r a role expression and C and D concept expressions.

Name	Syntax	Semantics
Inverse role	s^-	$\{(x,y) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid (y,x) \in s^\mathcal{I}\}$
Universal role	u	$\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
Top	$\top$	$\Delta^\mathcal{I}$
Bottom	$\bot$	$\emptyset$
Negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
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. restriction	$\forall r.C$	$\{x \mid \forall y.(x,y) \in r^\mathcal{I} \rightarrow y \in C^\mathcal{I}\}$
Exist. restriction	$\exists r.C$	$\{x \mid \exists y.(x,y) \in r^\mathcal{I} \wedge y \in C^\mathcal{I}\}$
Self concept	$\exists r.Self$	$\{x \mid (x,x) \in r^\mathcal{I}\}$
Qualified number	$\leqslant n\, r.C$	$\{x \mid \#\{y \in C^\mathcal{I} \mid (x,y) \in r^\mathcal{I}\} \leq n\}$
Restriction	$\geqslant n\, r.C$	$\{x \mid \#\{y \in C^\mathcal{I} \mid (x,y) \in r^\mathcal{I}\} \geq n\}$

Table 2. Syntax and semantics of $\mathcal{SROIQ}$ axioms.

Axiom α	$\mathcal{I} \models \alpha$, if	
$r_1 \circ \cdots \circ r_n \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$	RBox $\mathcal{R}$
$\mathsf{Dis}(s, r)$	$s^{\mathcal{I}} \cap r^{\mathcal{I}} = \emptyset$	
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	TBox $\mathcal{T}$
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	ABox $\mathcal{A}$
$r(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$	
$a \doteq b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$	
$a \not\doteq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$	

Answer-Set Programming. We review the basic notions of answer set programming [19] under the stable model semantics [8], for further details we refer to [4,7].

We fix a countable set $\mathcal{U}$ of *(domain) elements*, also called *constants*; and suppose a total order $<$ over the domain elements. An *atom* is an expression $p(t_1, \ldots, t_n)$, where p is a *predicate* of arity $n \geq 0$ and each t_i is either a variable or an element from $\mathcal{U}$. An atom is *ground* if it is free of variables. $B_{\mathcal{U}}$ denotes the set of all ground atoms over $\mathcal{U}$. A *(disjunctive) rule* ρ is of the form

$$a_1, \ldots, a_n \leftarrow b_1, \ldots, b_k, \ \mathit{not}\, b_{k+1}, \ldots, \ \mathit{not}\, b_m.$$

with $m \geq k \geq 0$, where $a_1, \ldots, a_n, b_1, \ldots, b_m$ are atoms, and "*not*" denotes *default negation*. The *head* of ρ is the set $H(\rho) = \{a_1, \ldots, a_n\}$ and the *body* of ρ is $B(\rho) = \{b_1, \ldots, b_k, \mathit{not}\, b_{k+1}, \ldots, \mathit{not}\, b_m\}$. Furthermore, $B^+(\rho) = \{b_1, \ldots, b_k\}$ and $B^-(\rho) = \{b_{k+1}, \ldots, b_m\}$. A rule ρ is *safe* if each variable in ρ occurs in $B^+(r)$. A rule ρ is *ground* if no variable occurs in ρ. A *fact* is a ground rule with empty body. An *(input) database* is a set of facts. A (disjunctive) *program* is a finite set of disjunctive rules. For a program Π and an input database D, we often write $\Pi(D)$ instead of $D \cup \Pi$. For any program Π, let U_Π be the set of all constants appearing in Π. $Gr(\Pi)$ is the set of rules $\rho\sigma$ obtained by applying, to each rule $\rho \in \Pi$, all possible substitutions σ from the variables in ρ to elements of U_Π.

An *interpretation* $I \subseteq B_{\mathcal{U}}$ *satisfies* a ground rule ρ iff $H(\rho) \cap I \neq \emptyset$ whenever $B^+(\rho) \subseteq I$, $B^-(\rho) \cap I = \emptyset$. I satisfies a ground program Π, if each $\rho \in \Pi$ is satisfied by I. A non-ground rule ρ (resp., a program Π) is satisfied by an interpretation I iff I satisfies all groundings of ρ (resp., $Gr(\Pi)$). $I \subseteq B_{\mathcal{U}}$ is an *answer set* (also called *stable model*) of Π iff it is a subset-minimal set satisfying the *Gelfond-Lifschitz reduct* $\Pi^I = \{H(\rho) \leftarrow B^+(\rho) \mid I \cap B^-(\rho) = \emptyset, \rho \in Gr(\Pi)\}$. For a program Π, we denote the set of its answer sets by $\mathcal{AS}(\Pi)$.

For a program Π, we denote the set of its answer sets by $\mathcal{AS}(\Pi)$, and might use $\mathcal{AS}(\Pi)_{|P}$ to project on the predicates $P = \{p_1, \ldots, p_n\}$.

We make use of further syntactic extensions, namely integrity constraints and count expressions, which both can be recast to ordinary normal rules as described in [7]. An *integrity constraint* is a rule ρ where $H(\rho) = \emptyset$, intuitively representing an undesirable situation; i.e. it has to be avoided that $B(\rho)$ evaluates positively. Count expressions are of the form $\#count\{l : l_1, \ldots, l_i\} \bowtie u$, where l is an atom and $l_j = p_j$ or $l_j = not\, p_j$, for p_j an atom, $1 \leq j \leq i$, u a non-negative integer, and $\bowtie \in \{\leq, <, =, >, \geq\}$. The expression $\{l : l_1, \ldots, l_n\}$ denotes the set of all ground instantiations of l, governed through $\{l_1, \ldots, l_n\}$. We restrict the occurrence of count expressions in a rule ρ to $B^+(\rho)$ only. Intuitively, an interpretation satisfies a count expression, if $N \bowtie u$ holds, where N is the cardinality of the set of ground instantiations of l, $N = |\{l \mid l_1, \ldots, l_n\}|$, for $\bowtie \in \{\leq, <, =, >, \geq\}$ and u a non-negative integer.

In order to handle (subset) preferences over answer-sets w.r.t. to ground instances of a specific atom, we make use of `asprin` [3]. The framework is designed to support and simplify the incorporation of preferences over answer-sets.

3 Justifications Under Fixed-Domain Semantics

3.1 Fixed-Domain Semantics

The standard semantics of DLs is defined on arbitrary domains. While *finite model reasoning* (a natural assumption in database theory) has become the focus of studies in DLs [5,18,25], where one is interested in models over arbitrary but finite domains, we consider the case where the domain has an *a-priori known* cardinality and use the term *fixed-domain*. This restriction yields an advantage regarding computational complexity for expressive DLs, but it also seems to reflect the intuitive model-theoretic expectations of practitioners in the industrial use cases we were confronted with. Satisfiability checking in $\mathcal{SROIQ}$ under the standard semantics is N2ExpTime-complete [16], while being NP-complete in the fixed-domain setting [6].

Definition 1 (Fixed-Domain Semantics [6]). *Let Δ be a non-empty finite set called fixed domain. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is said to be* Δ-fixed, *if $\Delta^{\mathcal{I}} = \Delta$, and $a^{\mathcal{I}} = a$ for all $a \in \Delta$. For a DL knowledge base $\mathcal{K}$, we call an interpretation $\mathcal{I}$ a* Δ-model *of $\mathcal{K}$ (and write $\mathcal{I} \models_\Delta \mathcal{K}$), if $\mathcal{I}$ is a Δ-fixed interpretation and $\mathcal{I} \models \mathcal{K}$. A knowledge base $\mathcal{K}$ is called* Δ-satisfiable, *if it has a Δ-model. A knowledge base is said to $\mathcal{K}$* Δ-entail *an axiom α ($\mathcal{K} \models_\Delta \alpha$), if $\mathcal{I} \models \alpha$ for every $\mathcal{I} \models_\Delta \mathcal{K}$.*

3.2 Justifications

Logical modeling is prone to error, and it is therefore important to provide debugging support. One of the most investigated methods is to determine explanations of certain entailments. These explanations are usually (minimal) subsets of the input knowledge base that suffice to entail the axiom in question. Several terms have been coined to refer to such (sub)sets. In the context of lightweight

description logics *Minimal Axiom Sets* (MinAs) is used, while the task of finding them is called Axiom Pinpointing [2,22]. Instead, for propositional logic, the term *Minimal Unsatisfiable Subformula* (MUS) to explain unsatisfiability was introduced long before [21]. In this paper we use the notion of *justification*, introduced in the context of highly expressive description logics [15].

Definition 2 (Justification [15]). *Let $\mathcal{K}$ be a knowledge base such that $\mathcal{K} \models \alpha$. $\mathcal{J}$ is a justification for α in $\mathcal{K}$ if $\mathcal{J} \subseteq \mathcal{K}$ and $\mathcal{J} \models \alpha$, and for all $\mathcal{J}' \subset \mathcal{J}, \mathcal{J}' \not\models \alpha$.*

Obviously, there may be multiple justifications for an axiom α. Dually to justifications, one might be interested in minimal subsets that can be retracted in order to restore consistency, or remove the unwanted entailment; commonly called *repair*. These two notions are strongly related in the sense that any repair has a non-empty intersection with each justification. However, in this work we restrict ourselves to justifications only.

Regarding the fixed-domain semantics, any justification needs to adhere to the considered fixed domain. Note that fixed-domain reasoning is monotonic, since otherwise, the subset minimality criterion in the definition of justifications would not be reasonable.

Definition 3 (Fixed-Justification). *Let $\mathcal{K}$ be a knowledge base, and Δ a fixed-domain such that $\mathcal{K} \models_\Delta \alpha$. $\mathcal{J}$ is a Δ-justification for α in $\mathcal{K}$ if $\mathcal{J} \subseteq \mathcal{K}$ and $\mathcal{J} \models_\Delta \alpha$, and for all $\mathcal{J}' \subset \mathcal{J}, \mathcal{J}' \not\models_\Delta \alpha$.*

It is the case that, if $\mathcal{K} \models \alpha$, then $\mathcal{K} \models_\Delta \alpha$ for any fixed-domain Δ. However, it does not hold that, if $\mathcal{J}$ is a justification for $\mathcal{K} \models \alpha$, then $\mathcal{J}$ is a Δ-justification for $\mathcal{K} \models_\Delta \alpha$ for any fixed-domain Δ. Due to a stronger restriction on models, there might exist $\mathcal{J}' \subset \mathcal{J}$, such that $\mathcal{J}' \not\models \alpha$ but $\mathcal{J}' \models_\Delta \alpha$. Nonetheless, giving a justification $\mathcal{J}$ under the standard semantics is helpful, since only subsets of $\mathcal{J}$ need to be considered. Formally, if $\mathcal{J}$ is a justification for $\mathcal{K} \models \alpha$, then there exist no Δ-justification $\mathcal{J}' \supset \mathcal{J}$ for $\mathcal{K} \models_\Delta \alpha$, for any fixed-domain Δ. This holds for any restricted reasoning maintaining monotonicity (e.g. finite model reasoning).

We focus on finding justifications for inconsistency, since entailment checking in $\mathcal{SROIQ}$ can be reduced to satisfiability checking. For example, $\mathcal{K} \models_\Delta A \sqsubseteq B$, iff $\mathcal{K} \cup \{(A \sqcap \neg B)(a)\}$ is Δ-inconsistent, where a is a fresh individual not occurring in $\mathcal{K}$. In the same way, justifications for entailments can be reduced to finding justifications for inconsistency. The caveat is that the introduced axiom should be fixed and not be part of candidate subset guessing.

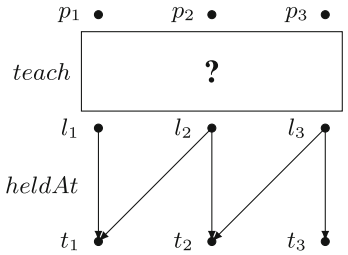
Example 1. We consider a simple assignment problem, encoded in $\mathcal{K}_{as}$. We let the domain be $\Delta = \{p_1, p_2, p_3, l_1, l_2, l_3, t_1, t_2, t_3\}$.

$$\begin{aligned}
Lecture \sqsubseteq \exists teach^{-}.Prof \quad Prof \sqsubseteq \leq 1\ teach.Lecture && (\alpha_{1-2})\\
SpecialLecture \sqsubseteq Lecture \quad SpecialLecture \sqsubseteq \forall teach^{-}.\{p_2\} && (\alpha_{3-4})\\
Lecture \sqsubseteq \neg Prof \quad Lecture \sqsubseteq \neg Time \quad Prof \sqsubseteq \neg Time && (\alpha_{5-7})\\
\exists heldAt \sqsubseteq Lecture \quad \top \sqsubseteq \forall heldAt.Time && (\alpha_{8-9})\\
teach \circ heldAt \sqsubseteq busyAt && (\alpha_{10})
\end{aligned}$$

First, we introduce the core of the knowledge base. Axioms α_{1-2} specify that a lecture must be taught by a professor, but one professor teaches at most one lecture. Axioms α_{3-4} introduce special lectures that can only be taught by professor p_2. Pairwise disjointness of the classes of lectures, professors and times is represented by axioms α_{5-7}. The domain and the range of *heldAt* are restricted by α_{8-9}. Finally, axiom α_{10} defines that a professor is busy at a certain time if he teaches a lecture at that time.

We specify the ABox for $\mathcal{K}_{as}$ in Fig. 1. As shown by the graph, this knowledge base is designed to find a suitable *teach* "configuration". Then, we add additional constraints $\neg busyAt(p_1, t_2)$ $[\alpha_{25}]$ and $\{p_3\} \sqsubseteq \leq 1\ busyAt.Time$ $[\alpha_{26}]$. It is easy to see that those constraints enforce p_1 and p_3 to teach l_1. However, l_1 is a special lecture that can only be taught by p_2. Consequently, $\mathcal{K}_{as}$ is inconsistent. Then, for example $\mathcal{J}_{as} = \mathcal{K}_{as} \setminus \{\alpha_{11-13}, \alpha_{15}, \alpha_{17-19}, \alpha_{21-22}\}$ is a Δ-justification for $\mathcal{K}_{as}$ inconsistency.

Note that some assertions can be concluded implicitly, i.e. using the axioms in $\mathcal{J}_{as}$, we can infer that p_1, p_2, p_3 must be professors since other elements in the domain are lectures and time points. Thus, we can remove them to get a minimal justification. Besides $\mathcal{J}_{as}$, there are 52 justifications in total. Also note that $\mathcal{K}_{as}$ is consistent under the standard semantics, since new professors can be introduced to teach problematic lectures.



$$\begin{aligned}
Prof(p_1)\quad Prof(p_2)\quad Prof(p_3) && (\alpha_{11-13})\\
Lecture(l_2)\quad Lecture(l_3) && (\alpha_{14-15})\\
SpecialLecture(l_1) && (\alpha_{16})\\
Time(t_1)\quad Time(t_2)\quad Time(t_3) && (\alpha_{17-19})\\
heldAt(l_1, t_1) && (\alpha_{20})\\
heldAt(l_2, t_1)\quad heldAt(l_2, t_2) && (\alpha_{21-22})\\
heldAt(l_3, t_2)\quad heldAt(l_3, t_3) && (\alpha_{23-24})
\end{aligned}$$

Fig. 1. $\mathcal{K}_{as}$ ABox representation and axioms.

4 Computing Justifications

Algorithms for finding justifications can be categorized coarsely into *black-box* and *glass-box* approaches. *Black-box* approaches use a reasoner to conduct the reasoning tasks it was designed for, i.e. entailment checking. Contrarily, in the *glass-box* approach, reasoners are modified, i.e. the internal reasoning algorithms are tweaked towards justifications. Generally, black-box approaches are more robust and easier to implement, whereas the glass-box approaches provide more potential for optimization. Subsequently, we introduce two black-box approaches, followed by a dedicated glass-box approach.

4.1 Black-Box Approaches

The ontology editor Protégé, has built-in functionality to compute justifications under the standard semantics, which is based on the OWL Explanation Workbench[2] [12]. The underlying algorithm is based on the Hitting-Set Tree (HST) algorithm originating from Reiter's theory of diagnosis [24]. For the details of the implementation we refer to [11].

Axiomatization of Δ-models. Given a knowledge base $\mathcal{K}$ and a fixed domain $\Delta = \{a_1, \ldots, a_n\}$, one can axiomatize the fixed-domain semantics, such that $\mathcal{K} \models_\Delta \alpha$ iff $\mathcal{K} \cup \mathcal{FD}_\Delta \models \alpha$, where $\mathcal{FD}_\Delta = \{\top \sqsubseteq \{a_1, \ldots a_n\}\} \cup \{a_i \not\approx a_j \mid 1 \leq i < j \leq n\}$. It is easy to see, that those axioms enforce reasoning over Δ. A black-box algorithm for finding justifications merely exploits inconsistency or entailment checking, which is a standard reasoning task, thus standard DL reasoners can be used for fixed-domain standard reasoning. In Sect. 5 we will therefore use the explanation workbench with `HermiT` as black-box reasoner.

A Fixed-Domain Reasoner as a Black-box. `Wolpertinger` has been introduced as reasoner adhering to the fixed-domain semantics [28], which can easily be plugged into the explanation workbench. We will evaluate the performance of this approach, and expect the performance to correlate with the performance of entailment checking. With `W-black-box` we refer to this approach in the subsequent evaluation.

4.2 A Glass-Box Approach Using Answer-Set Programming

We now introduce a glass-box approach for computing justifications using an encoding into answer-set programming. The translation is based on the naïve translation [6], which has already been implemented in `Wolpertinger`, but some fundamental changes needed to be made in order to compute justifications. Since finding justifications is about finding the corresponding (minimal) subsets of a knowledge base, another "layer" is required, on the top of the model correspondence established in the naïve translation, which is not straightforward to encode

[2] Subsequently just called *explanation workbench*.

in ASP. We will therefore avoid negation-as-failure, and hence refer to this new translation as *naff* (negation-as-failure free). Subsequently, the translation is depicted in detail.

Let $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ be a normalized $\mathcal{SROIQ}$ knowledge base, and Δ a fixed domain.[3] With $\Pi(\mathcal{K}, \Delta) = \Pi_{gen}(\mathcal{K}, \Delta) \cup \Pi_{chk}(\mathcal{K}) \cup \Pi_{inc}(\mathcal{K})$, we denote the translation of $\mathcal{K}$ into a logic program w.r.t. Δ. Intuitively, $\Pi_{gen}(\mathcal{K}, \Delta)$ generates candidate interpretations w.r.t. Δ, and each axiom is translated into rules in $\Pi_{chk}(\mathcal{K})$, in such a way, that any violation will cause a dedicated propositional atom to be true. If so, the *Principle of Explosion* (POE) is applied via appropriate rules. For every translated axiom, an additional dedicated propositional activator is added in the body of the resulting rule, allowing to activate or deactivate the rule, thus indicating whether to include or exclude the axiom in a candidate justification. With the disjunctive rules in $\Pi_{gss}(\mathcal{K}, \Delta)$, the generation of extensions for every concept and role name is realized.

$$\begin{aligned}\Pi_{gss}(\mathcal{K}, \Delta) = \{&A(X), not\_A(X) :- \top(X) \mid A \in N_C(\mathcal{K})\} \cup \\ \{&r(X,Y), not\_r(X,Y) :- \top(X), \top(Y) \mid r \in N_R(\mathcal{K}\} \cup \\ \{&\top(a) \mid a \in \Delta\}.\end{aligned}$$

Atomic clashes need to be detected explicitly, which is done via simple rules in $\Pi_{obv}(\mathcal{K})$. Note that clashes are not represented by constraints, but rules with the dedicated propositional variable *inc* as head.

$$\begin{aligned}\Pi_{obv}(\mathcal{K}) = \{&inc :- A(X), not\_A(X) \mid A \in N_C(\mathcal{K})\} \cup \\ \{&inc :- r(X,Y), not\_r(X,Y) \mid r \in N_R(\mathcal{K}\}.\end{aligned}$$

Based on the detection of atomic clashes, the rules in $\Pi_{poe}(\mathcal{K})$ encode the POE, that is, every concept and role assertion follows whenever *inc* holds.

$$\begin{aligned}\Pi_{poe}(\mathcal{K}) = \{&A(X) :- inc, \top(X) \mid A \in N_C(\mathcal{K})\} \cup \\ \{&not\_A(X) :- inc, \top(X) \mid A \in N_C(\mathcal{K})\} \cup \\ \{&r(X,Y) :- inc, \top(X), \top(Y) \mid r \in N_R(\mathcal{K}\} \cup \\ \{&not\_r(X,Y) :- inc, \top(X), \top(Y) \mid r \in N_R(\mathcal{K}\}.\end{aligned}$$

Qualified Number Restriction Encoding. One problem that we encountered is the usage of the $<$-operator in the translation of at-least cardinality restrictions. Consider the concept $\geq n\ r.C$, which restricts an individual to have at least n r-neighbors, that are a member of C. The intuitive translation is a constraint that counts how many outgoing r-connections exist, satisfying also the membership in C, thus failing if there are less than n r-neighbors not satisfying the condition. However, this translation does not work anymore due to the rules in $\Pi_{poe}(\mathcal{K})$.

[3] We do not provide details on the normalization part, an refer instead to our previous work [6].

We therefore introduce a different view of the semantics of cardinality restrictions in the fixed-domain setting. For simplicity, we define $r.C(a) = \{x \in C^{\mathcal{I}} \mid (a,x) \in r^{\mathcal{I}}\}$. Hence $r.C(a)$ consists of all members of concept C that are connected via r starting in a. The idea is to count individuals which are not a member of the concept where this restriction applies. There are two possibilities that an individual b is not in $r.C(a)$: $(a,b) \notin r$ or $b \notin C$. Let $n = |\Delta^{\mathcal{I}}|$ and $m = |\{b \in \Delta^{\mathcal{I}} \mid b \notin r.C(a)\}|$. Hence, the number of individuals in $r.C(a)$ is $n - m$. This is only possible due to the given fixed domain.

Proposition 1. *Let $\mathcal{K}$ be a $\mathcal{SROIQ}$ knowledge base, Δ be a fixed-domain, and $\mathcal{I}$ a Δ-model of $\mathcal{K}$. Then $(\geq n\ r.C)^{\mathcal{I}} = \{x \in \Delta \mid \#\{y \in \Delta \mid y \notin C^{\mathcal{I}} \text{ or } (x,y) \notin r^{\mathcal{I}}\} \leq |\Delta| - n\}$.*

Hence, we can compute such a relation between two individuals to be used later in the translation of axioms. A new auxiliary predicate is introduced for each pair of concept (and its negation) and role. We define:

$$\begin{aligned}\Pi_{nra}(\mathcal{K}) = {} & \{not\_r\_C(X,Y) :\!- \ not\_C(Y) \mid C \in N_C(\mathcal{K}), r \in N_R(\mathcal{K})\} \cup \\ & \{not\_r\_C(X,Y) :\!- \ not\_r(X,Y) \mid C \in N_C(\mathcal{K}), r \in N_R(\mathcal{K})\} \cup \\ & \{not\_r\_not\_C(X,Y) :\!- \ C(Y) \mid C \in N_C(\mathcal{K}), r \in N_R(\mathcal{K})\} \cup \\ & \{not\_r\_not\_C(X,Y) :\!- \ not\_r(X,Y) \mid C \in N_C(\mathcal{K}), r \in N_R(\mathcal{K})\}.\end{aligned}$$

$\Pi_{nra}(\mathcal{K})$ does not change the interpretation built by $\Pi_{gen}(\mathcal{K})$. It merely collects all those individuals satisfying the previously mentioned conditions. Additionally, we have to take care about inverse roles, for which the rules look similar, but variables need to be swapped. Finally, $\Pi_{gen}(\mathcal{K}, \Delta) = \Pi_{gss}(\mathcal{K}, \Delta) \cup \Pi_{obv}(\mathcal{K}) \cup \Pi_{poe}(\mathcal{K}) \cup \Pi_{nra}(\mathcal{K})$.

ABox Translation. The first pruning of the search space originates from ABox assertions. As the input is a normalized knowledge base, each assertion contains only a literal concept, or literal role, respectively. It then straightforward to encode:

$$\begin{aligned}\Pi_{chk}(\mathcal{A}) = {} & \{inc :\!- \ active(i), not\_A(a) \mid A(a) \in \mathcal{A}\} \cup \\ & \{inc :\!- \ active(i), A(a) \mid \neg A(a) \in \mathcal{A}\} \cup \\ & \{inc :\!- \ active(i), not\_r(a,b) \mid r(a,b) \in \mathcal{A}\} \cup \\ & \{inc :\!- \ active(i), r(a,b) \mid \neg r(a,b) \in \mathcal{A}\}.\end{aligned}$$

TBox Translation. Each TBox axiom is normalized and of form $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$, with each C_i being non-complex, i.e. one of the concept constructors depicted in Table 3. It is then easy to turn normalized axioms into appropriate rules to detect any violation.

$$\Pi_{chk}(\mathcal{T}) = \{inc :\!- \ active(j), \tau(C_1), ..., \tau(C_n), \top(X) \mid \top \sqsubseteq \bigsqcup_{i=1}^{n} C_i \in \mathcal{T}\}$$

Table 3. Translation of concept constructors. Note: O_a is a new concept name unique for a, and $m = |\Delta^{\mathcal{I}}|$.

C	$\tau(C)$
A	$not\_A(X)$
$\neg A$	$A(X)$
$\{a\}$	$\{not\_O_a(X)\}, \{O_a(a)\}$
$\forall r.A$	$\{not\_A(Y), r(X,Y)\}$
$\forall r.\neg A$	$\{A(Y), r(X,Y)\}$
$\exists r.Self$	$not\_r(r,X,X)$
$\neg\exists r.Self$	$r(r,X,X)$
$\geq n\ r.A$	$\#count\{Y : not\_r\_A(X,Y)\} > (m-n)$
$\geq n\ r.\neg A$	$\#count\{Y : not\_r\_not\_A(X,Y)\} > (m-n)$
$\leq n\ r.A$	$\#count\{Y : r(X,Y), \mathrm{A}(Y)\} > n$
$\leq n\ r.\neg A$	$\#count\{Y : r(X,Y), \mathrm{not\_A}(Y)\} > n$

RBox Translation. Since normalized, each axiom in an RBox $\mathcal{R}$ is either a (simplified) role chain, disjointness or role inclusion axiom. As for TBox axioms, each axiom in $\mathcal{R}$ is translated into a rule that enforces the propositional variable *inc* to be true, whenever the axiom is violated.

$$\begin{aligned}\Pi_{chk}(\mathcal{R}) = \{&inc :\!- \ active(i), r(X,Y),\ s(X,Y) \mid \mathsf{Dis}(r,s) \in \mathcal{R}\} \cup \\ \{&inc :\!- \ active(i), r(X,Y),\ not\_s(X,Y) \mid r \sqsubseteq s \in \mathcal{R}\} \cup \\ \{&inc :\!- \ active(i), s_1(X,Y), s_2(Y,Z), not\_r(X,Z) \mid s_1 \circ s_2 \sqsubseteq r \in \mathcal{R}\}.\end{aligned}$$

For example, α_2 and α_{10} in Example 1 are encoded as:

```
inc :- active(2), prof(X), #count{Y: teach(X,Y), lecture(Y)}>1.
inc :- active(10), teach(X,Y), heldAt(Y,Z), not_busyAt(X,Z).
```

Finally, let $\Pi_{chk}(\mathcal{K}) = \Pi_{chk}(\mathcal{A}) \cup \Pi_{chk}(\mathcal{T}) \cup \Pi_{chk}(\mathcal{R})$, be the translation of all axioms in a knowledge base. It remains to remove any candidate answer-set not including *inc*, as well as guessing the set of active rules. As a result, any answer-set now indicates which axioms jointly cause the inconsistency. Then, preferring answer-sets that are subset-minimal w.r.t. the set of ground instances of *active* yield exactly the desired justifications. The following program captures these requirements and completes the translation $\Pi(\mathcal{K}, \Delta) = \Pi_{gen}(\mathcal{K}, \Delta) \cup \Pi_{chk}(\mathcal{K}) \cup \Pi_{inc}(\mathcal{K})$.

$$\begin{aligned}\Pi_{inc}(\mathcal{K}) = \{ &:\!- \ not\ inc. \\ &\{active(X) :\!- \ X = 1..n\}. \\ &\#optimize(p). \\ &\#preference(p, subset)\{active(C) : C = 1..n\}.\ \}\end{aligned}$$

Theorem 1. *Let* $ACT(\mathcal{K}) = \{active(1), \ldots, active(n)\}$, *where* $n = |\mathcal{K}|$ *and* $\mathcal{K}^X = \{\alpha_i \in \mathcal{K} \mid active(i) \in X\}$. *Then* $\mathcal{AS}(\Pi(\mathcal{K}, \Delta))_{|\{active\}} = \{X \in 2^{ACT(\mathcal{K})} \mid \mathcal{K}^X$ *is* Δ*-inconsistent*$\}$.

Proof sketch. Using the well-known splitting theorem [17], we split $\Pi(\mathcal{K}, \Delta)$ into two parts: axiom (subset) guessing and inconsistency checking. First, we show that each X representing a potential subset can be used to reduce the program to $\Pi(\mathcal{K}^X, \Delta)$. For the second part, we show that if $\mathcal{K}^X$ is Δ-inconsistent, $\mathcal{AS}(\Pi(\mathcal{K}^X), \Delta)$ consists only of exactly one answer set. Combining both arguments via the splitting theorem, it can be concluded that each answer set of $\Pi(\mathcal{K}, \Delta)$ corresponds to a Δ-justification for inconsistency of $\mathcal{K}$.□

We implemented this glass-box approach into `Wolpertinger`. In the evaluation, we refer to this approach as `W-glass-box`. While our translated programs need to be evaluated by `asprin` (which needs `Clingo`), it would be easy to remove the minimality preference, such that each answer set then corresponds to an inconsistent subset of the knowledge base. One could the also define (other) preferences, e.g. prioritizing some axioms to be necessarily included.

5 Evaluation

We introduce several simple constraint-type combinatorial problems that are aligned with our approach. We deliberately make them inconsistent, with a controlled number of justifications. The evaluations were performed on an HPC system with Haswell CPUs using a 4 GB memory limit. Unless stated differently, the timeout for each evaluation was 30 min. We use the hyphen symbol (-) to denote a *timeout*.

We reused an unsatisfiable knowledge base described in [6]. The knowledge base represents a Pigeonhole-type problem. We specified the axioms such that we want a model that depicts an r-chain of length $n+1$, but fixed the domain to n elements, for which a model cannot exist. For $\mathcal{K}_n = (\mathcal{T}_n, \mathcal{A}_n)$, we have:

$$\begin{aligned}\mathcal{T}_n &= \{A_1 \sqsubseteq \exists r.A_2, \ldots,\ A_n \sqsubseteq \exists r.A_{n+1}\} \cup \\ &\qquad \{A_i \sqcap A_j \sqsubseteq \bot \mid 1 \leq i < j \leq n+1\} \\ \mathcal{A}_n &= \{A_1(a_1)\} \\ \Delta_n &= \{a_1, ..., a_n\}\end{aligned}$$

First, a comparison between the naïve and naff translation as been made. We expected the naff translation to be somewhat slower due to the overhead of computing some auxiliary atoms, which is confirmed as depicted in Table 4 (left). Afterwards, the performance of each approach to compute (Δ)-justifications (of inconsistency) of this knowledge base has been evaluated. In this case, since the only justification is the whole knowledge base, there is no major difference between requesting only one, or all justifications. This would be different if the only justification is a proper subset, because the algorithm has to make sure there

Table 4. Runtimes for checking unsatisfiability of $\mathcal{K}_n$ (left table), and runtimes of each approach for computing one justification.

#	Instance	naff	naïve
1	$\mathcal{K}_5$	0.013 s	0.010 s
2	$\mathcal{K}_6$	0.042 s	0.025 s
3	$\mathcal{K}_7$	0.092 s	0.063 s
4	$\mathcal{K}_8$	0.429 s	0.320 s
5	$\mathcal{K}_9$	5.324 s	3.805 s
6	$\mathcal{K}_{10}$	105.202 s	78.208 s
7	$\mathcal{K}_{11}$	–	1 423.463 s
8	$\mathcal{K}_{12}$	–	–

#	Instances	`H-black-box`	`W-black-box`	`W-glass-box`
1	$\mathcal{K}_5$	6.212 s	6.742 s	0.207 s
2	$\mathcal{K}_6$	7.023 s	6.284 s	0.277 s
3	$\mathcal{K}_7$	8.197 s	7.735 s	0.352 s
4	$\mathcal{K}_8$	9.521 s	9.057 s	2.510 s
5	$\mathcal{K}_9$	25.752 s	23.959 s	17.397 s
6	$\mathcal{K}_{10}$	206.457 s	518.377 s	–
7	$\mathcal{K}_{11}$	2 274.480 s	–	–

is no other justification. Table 4 (right) shows the result. It can be stressed, that for smaller instances, the `W-glass-box` approach performs best, followed by `W-black-box`. However, they do not scale well for bigger instances where `H-black-box` outperforms both of them. For the latter experiment, the timeout was set to one hour.

The second knowledge base $\mathcal{K}_{(m,n)}$ used for evaluation heavily uses cardinality restrictions. Individuals of the source concept C need to be connected with at least n individuals that are each a member of the concept A_i, where $1 \leq i \leq m$. However, we restrict the domain to contain only $n+1$ elements. Finally, we impose a constraint such that all concepts are disjoint. Obviously, the existence of two such axioms already cause an inconsistency. For $\mathcal{K}_{(m,n)} = (\mathcal{T}_{(m,n)}, \mathcal{A})$ we have:

$$\begin{aligned}
\mathcal{T}_{(m,n)} = {} & \{C \sqsubseteq \geq n\ r.A_1, \ldots, C \sqsubseteq \geq n\ r.A_m\} \cup \\
& \{C \sqsubseteq \neg A_1, \ldots, C \sqsubseteq \neg A_m\} \cup \\
& \{A_1 \sqsubseteq \neg A_2, A_1 \sqsubseteq \neg A_3, \ldots, A_{m-1} \sqsubseteq \neg A_m\} \\
\mathcal{A} = {} & \{C(a)\} \\
\Delta_n = {} & \{a, x_1, ..., x_n\}
\end{aligned}$$

The result is shown in Table 5. The black-box approach with HermiT failed to compute the justifications for any case within the time limit. This result indicates that standard reasoners struggle in handling cardinality restrictions under the fixed-domain semantics. We suppose that the result originates from the fact that $\geq$-cardinality is handled easily in standard semantics since the reasoner can introduce new individuals satisfying the restriction. While `H-black-box` is able to solve some of the instances, `W-glass-box` computes all of them in reasonable time.

The third evaluation is based on the graph-coloring problem. We encode some instances of the Mycielskian graphs[4]. Since the chromatic number of each instance is provided, making them non-colorable is trivial. For a graph with chromatic number n, we only provide $n-1$ colors. The result is shown in Table 6. Each approach exceeded the timeout for the larger instances. Similar to the cardinality evaluation, the `W-glass-box` approach performs best. For the

[4] http://mat.gsia.cmu.edu/COLOR/instances.html.

Table 5. Runtime for individual cardinality.

#	Instances	`H-black-box`	`W-black-box`	`W-glass-box`
1	$\mathcal{K}_{10,10}$	–	94.787 s	3.461 s
2	$\mathcal{K}_{10,20}$	–	75.107 s	5.141 s
3	$\mathcal{K}_{10,30}$	–	104.382 s	8.029 s
4	$\mathcal{K}_{20,10}$	–	448.757 s	45.578 s
5	$\mathcal{K}_{20,20}$	–	–	66.123 s
6	$\mathcal{K}_{20,30}$	–	–	103.721 s
7	$\mathcal{K}_{30,10}$	–	634.572 s	331.576 s
8	$\mathcal{K}_{30,20}$	–	–	476.985 s
9	$\mathcal{K}_{30,30}$	–	–	548.865 s

Table 6. Runtime for n-coloring problems.

#	Instances	#Nodes	#Edges	`H-black-box`	`W-black-box`	`W-glass-box`
1	$\mathcal{K}_{myciel3}$	11	20	43.335 s	71.347 s	1.423 s
2	$\mathcal{K}_{myciel4}$	23	71	–	–	11.327 s
3	$\mathcal{K}_{myciel5}$	47	236	–	–	–

small instance, `H-black-box` performs better than `W-black-box`. For the second instance, we find that `H-black-box` provided merely one justification before the timeout, while `W-black-box` was able to compute at least five justifications.

As shown in Table 4, `H-black-box` performs better in some cases. While finding justifications is a hard problem, asking for several of them is more feasible. The necessary adjustments can easily be done for each tool. Another important note to mention is, we only use one thread for evaluation, though the problem itself could be done in parallel.

6 Conclusion

We considered the task of computing justifications for entailments under the fixed-domain semantics, a task of general high importance in the modeling phase of ontologies. We proposed three different approaches to this problem and comparatively evaluated one using standard OWL technology employing an axiomatization of the fixed-domain semantics, one using our dedicated fixed-domain reasoner `Wolpertinger` in combination with Protégé's explanation workbench, and one where the problem is encoded entirely into answer-set programming. The evaluation suggests that each of the proposed approaches do have their difficulties as well as individual advantages. Hence, it remains imperative to conduct more experiments with different setups. Also, all tools were used in their standard configuration, which gives another optimization angle.

Moreover, other approaches developed to debug answer-set programs need to be considered and compared. For example, Pontelli et al. suggest a method

to obtain justifications for the truth value of an atom in an answer-set [23], which might be reused in our setting to obtain an explanation for inconsistency (represented by the propositional atom *inc*). A different approach is the stepwise debugging methodology proposed by Oetsch et al. which allows to identify rules that prohibit a certain (partial) answer-set [20]. However, this approach is designed to answer the question, why some interpretation is actually not an answer-set of the program, thus we see it as future work to identify how this approach can be resembled into our setting. Moreover, it would be a great feature for users if a tool actually recommended automatic repairs in addition to the justifications, which might be realized be using these related approaches.

Acknowledgements. We are grateful for the valuable feedback from the anonymous reviewers, which helped greatly to improve this work. This work is supported by DFG in the Research Training Group QuantLA (GRK 1763) and in the Research Training Group RoSI (GRK 1907).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation, and Applications, 2nd edn., Cambridge University Press (2007)
2. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic $\mathcal{EL}^+$. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) KI 2007. LNCS (LNAI), vol. 4667, pp. 52–67. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74565-5_7
3. Brewka, G., Delgrande, J.P., Romero, J., Schaub, T.: asprin: Customizing answer set preferences without a headache. In: AAAI, pp. 1467–1474. AAAI Press (2015)
4. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011)
5. Calvanese, D.: Finite model reasoning in description logics. In: Proceedings of the 5th International Conference on the Principles of Knowledge Representation and Reasoning (KR 1996), pp. 292–303. Morgan Kaufmann (1996)
6. Gaggl, S.A., Rudolph, S., Schweizer, L.: Fixed-domain reasoning for description logics. In: Kaminka, G.A., et al. (eds.) Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016), Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 819–827. IOS Press, September 2016
7. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer set solving in practice. Synth. Lect. Artif. Intell. Mach. Learn. **6**, 1–238 (2012)
8. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Gener. Comput. **9**(3/4), 365–386 (1991)
9. Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (eds.): Ontology Engineering with Ontology Design Patterns - Foundations and Applications, Studies on the Semantic Web, vol. 25. IOS Press (2016)
10. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation
11. Horridge, M.: Justification based explanation in ontologies. Ph.D. thesis, University of Manchester (2011)

12. Horridge, M., Parsia, B., Sattler, U.: Explanation of OWL entailments in Protege 4. In: Bizer, C., Joshi, A. (eds.) Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC 2008), 28 October 2008, CEUR Workshop Proceedings, vol. 401. CEUR-WS.org (2008)
13. Horridge, M., Parsia, B., Sattler, U.: Explaining inconsistencies in OWL ontologies. In: Godo, L., Pugliese, A. (eds.) SUM 2009. LNCS (LNAI), vol. 5785, pp. 124–137. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04388-8_11
14. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning, KR 2006, pp. 57–67. AAAI Press (2006)
15. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_20
16. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Brewka, G., Lang, J. (eds.) Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 274–284. AAAI Press (2008)
17. Lifschitz, V., Turner, H.: Splitting a logic program. In: ICLP, vol. 94, pp. 23–37 (1994)
18. Lutz, C., Sattler, U., Tendera, L.: The complexity of finite model reasoning in description logics. Inf. Comput. **199**(1–2), 132–171 (2005)
19. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Ann. Math. Artif. Intell. **25**(3–4), 241–273 (1999)
20. Oetsch, J., Pührer, J., Tompits, H.: Stepwise debugging of answer-set programs. TPLP **18**(1), 30–80 (2018)
21. Papadimitriou, C.H., Wolfe, D.: The complexity of facets resolved. J. Comput. Syst. Sci. **37**(1), 2–13 (1988)
22. Peñaloza, R., Sertkaya, B.: Understanding the complexity of axiom pinpointing in lightweight description logics. Artif. Intell. **250**, 80–104 (2017)
23. Pontelli, E., Son, T.C., El-Khatib, O.: Justifications for logic programs under answer set semantics. TPLP **9**(1), 1–56 (2009)
24. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1), 57–95 (1987)
25. Rosati, R.: Finite model reasoning in *DL-Lite*. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 215–229. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68234-9_18
26. Rudolph, S.: Foundations of description logics. In: Polleres, A., et al. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 76–136. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23032-5_2
27. Rudolph, S., Schweizer, L.: Not too big, not too small... complexities of fixed-domain reasoning in first-order and description logics. In: Oliveira, E., Gama, J., Vale, Z., Lopes Cardoso, H. (eds.) EPIA 2017. LNCS (LNAI), vol. 10423, pp. 695–708. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65340-2_57
28. Rudolph, S., Schweizer, L., Tirtarasa, S.: Wolpertinger: a fixed-domain reasoner. In: Nikitina, N., Song, D., Fokoue, A., Haase, P. (eds.) Proceedings of the ISWC 2017 Posters and Demonstrations and Industry Tracks Co-located with 16th International Semantic Web Conference (ISWC 2017), 23–25 October 2017, CEUR Workshop Proceedings, vol. 1963. CEUR-WS.org (2017)