ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-819

Fixed-Domain Reasoning for Description Logics

Sarah Gaggl and **Sebastian Rudolph** and **Lukas Schweizer**[1]

Abstract. After decades of fruitful research, description logics (DLs) have evolved into a de facto standard in logic-based knowledge representation. In particular, they serve as the formal basis of the standardized and very popular web ontology language (OWL), which also comes with the advantage of readily available user-friendly modeling tools and optimized reasoning engines. In the course of the wide-spread adoption of OWL and DLs, situations have been observed where logically less skilled practitioners are (ab)using these formalisms as constraint languages adopting a closed-world assumption, contrary to the open-world semantics imposed by the classical definitions and the standards. To provide a clear theoretical basis and inferencing support for this often practically reasonable "off-label use" we propose an alternative formal semantics reflecting the intuitive understanding of such scenarios. To that end, we introduce the *fixed-domain semantics* and argue that this semantics gives rise to an interesting new inferencing task: *model enumeration*. We describe how the new semantics can be axiomatized in very expressive DLs. We thoroughly investigate the complexities for standard reasoning as well as query answering under the fixed-domain semantics for a wide range of DLs. Further, we present an implementation of a fixed-domain DL reasoner based on a translation into answer set programming (ASP) which is competitive with alternative approaches for standard reasoning tasks and provides the added functionality of model enumeration.

1 Introduction

IT practitioners facing the task of developing logic-based specifications tend to prefer knowledge representation formalisms which are standardized, widely adopted and that come with elaborate modeling tool support. A notable example for this is the Web Ontology Language OWL [32]. Ontology editors like Protégé [16] provide user-friendly interfaces and intuitive access to a complex and involved formalism.

This leads to situations where OWL is chosen over other formalisms, even if the application scenario does not match the typical usage of this language. For example, problems of a constraint-satisfaction type do not go well with OWL's standard semantics allowing for models of arbitrary size. Consider graph 3-coloring as a short but representative constraint-satisfaction problem. It is easy to create some OWL axioms specifying the conditions on valid colorings of a given graph. Asking for colorability as such can then be cast into an OWL consistency problem, a natural task for OWL reasoners. Yet, generating (all) concrete colorings already requires capabilities that an OWL reasoner cannot provide out of the box, namely that of extracting or enumerating models.

To overcome these shortcomings, we propose *fixed-domain reasoning* for DLs – a family of logics providing the logical underpinning of OWL and its sublanguages. This intuitive and simple approach considers DLs under a non-standard model-theoretic semantics, modifying the modelhood condition by restricting the domain to an explicitly given, fixed finite set. We investigate the combined complexity of reasoning in the presence of a given fixed domain for a wide range of description logics, for which we establish tight bounds for standard reasoning tasks as well as query answering for various query notions. While satisfiability checking in OWL under the classical semantics is N2ExpTime-complete [15] and query answering is not even known to be decidable, we show that these problems under the *fixed-domain semantics* are merely NP-complete and Π_2^P-complete, respectively.

We note that the fixed-domain condition can be axiomatized in OWL. Still, employing the axiomatization and using available OWL reasoners would not allow for the non-standard reasoning task of model enumeration. Therefore, we propose a different approach and define a translation of $\mathcal{SROIQ}$ knowledge bases (the logical counterparts to OWL ontologies) into answer set programming (ASP, [4]), such that there is a one-to-one correspondence between the fixed-domain models of the considered knowledge base and the set of answer-sets of the obtained program. This allows us to use existing ASP solvers (see [5] for an overview) for fixed-domain reasoning – including standard as well as non-standard tasks. For the proposed translation, we provide an implementation and present preliminary evaluations on typical constraint-satisfaction-type problems. This not only demonstrates feasibility, but also suggests significant improvement compared to the axiomatized approach using highly optimized OWL reasoners.

2 Preliminaries

In this section, we provide the necessary background of description logics, Datalog queries, and answer-set programming. Despite obvious structural commonalities of Datalog queries and ASP, we introduce and treat them separately, since their purpose is different: the former is used as a query formalism over DL knowledge bases while the latter serves as the target formalism for a compilation process.

[1] Technische Universität Dresden, Computational Logic Group, e-mail: `firstname.lastname@tu-dresden.de`

Description Logics OWL 2 DL, the version of the Web Ontology Language we focus on, is defined based on description logics (DLs, [2, 26]). We briefly recap the description logic $\mathcal{SROIQ}$ (for details see [13]). 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 a $\mathcal{SROIQ}$ ABox, $\mathcal{T}$ is a $\mathcal{SROIQ}$ TBox and $\mathcal{R}$ is a $\mathcal{SROIQ}$ RBox. Table 2 presents the respective axiom types available in the three parts.[2] 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.

Table 1. Syntax and semantics of role and concept constructors in $\mathcal{SROIQ}$, where $a_1, \dots a_n$ 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	$\{a_1, \dots, a_n\}$	$\{a_1^{\mathcal{I}}, \dots, a_n^{\mathcal{I}}\}$
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	
$r_1 \circ \dots \circ r_n \sqsubseteq r$	$r_1^{\mathcal{I}} \circ \dots \circ r_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$	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 $\mathcal{A}$
$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$	
$a \doteq b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$	
$a \neq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$	

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.

[2] The original definition of $\mathcal{SROIQ}$ contained more RBox axioms (expressing transitivity, (a)symmetry, (ir)reflexivity of roles), but these can be shown to be syntactic sugar. Moreover, the definition of $\mathcal{SROIQ}$ contains so-called *global restrictions* which prevents certain axioms from occurring together. These complicated restrictions, while crucial for the decidability of classical reasoning in $\mathcal{SROIQ}$ are not necessary for fixed-domain reasoning considered here, hence we omit them for the sake of brevity.

Boolean Datalog Queries Here we briefly introduce syntax and semantics of Datalog queries over description logic knowledge bases. A *term* can be a variable from a countably infinite set V of variables, or an element of N_I. An *atom* has the form $p(t_1, \dots, t_n)$ where $t_1, \dots, t_n$ are terms and p is a predicate of arity n from a set Π of predicates containing N_C (arity 1) and N_R (arity 2) and containing a special predicate *goal* of arity 0. A *Boolean Datalog query* is a set of first-order logic Horn rules of the form $\forall X.a_1 \wedge \dots \wedge a_k \rightarrow a$ where $a_1, \dots, a_n, a$ are atoms, but the predicate of a is not from N_C or N_R. $X \subseteq V$ denotes the set of variables occurring in the atoms. Given a DL interpretation $\mathcal{I}$, and a Boolean Datalog query Q, an *extended model* for $\mathcal{I}$ and Q is a first-order interpretation $\mathcal{J}$ over $\Delta^{\mathcal{I}}$ that coincides with $\mathcal{I}$ on the interpretation of N_I, N_C, and N_R and satisfies all the rules from Q. We say that Q *matches* $\mathcal{I}$ and write $\mathcal{I} \models Q$ if $\mathcal{J} \models \mathit{goal}$ for every extended model $\mathcal{J}$ for $\mathcal{I}$ and Q. For a DL knowledge base $\mathcal{K}$, we say $\mathcal{K}$ *entails* Q iff $\mathcal{I} \models Q$ for every model $\mathcal{I}$ of $\mathcal{K}$. *Bounded arity Datalog queries* are classes of queries where the arity of the used predicates is bounded by some constant. A *Boolean conjunctive query* is a Boolean Datalog query with just one rule where $a_1, \dots, a_n$ use only predicates from $N_C \cup N_R$ and $a = \mathit{goal}$. In that case, such a query can be equivalently written as the first-order formula $\exists X.a_1 \wedge \dots \wedge a_k$.

Answer-Set Programming We review the basic notions of answer set programming [22] under the stable model semantics [11], for further details we refer to [4, 8].

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 $p(t_1, \dots, t_n)$, 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 *(normal) rule* ρ is of the form

$$a \leftarrow b_1, \dots, b_k,\ \mathit{not}\, b_{k+1}, \dots,\ \mathit{not}\, b_m.$$

with $m \geq k \geq 0$, where $a, b_1, \dots, b_m$ are atoms, and "*not*" denotes *default negation*. The *head* of ρ is the singleton set $H(\rho) = \{a\}$ and the *body* of ρ is $B(\rho) = \{b_1, \dots, b_k, \mathit{not}\, b_{k+1}, \dots, \mathit{not}\, b_m\}$. Furthermore, $B^+(\rho) = \{b_1, \dots, b_k\}$ and $B^-(\rho) = \{b_{k+1}, \dots, b_m\}$. 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 (normal) *program* is a finite set of normal 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 the 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 $\mathcal{AS}(\Pi)$.

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 [8]. 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.

3 Models over Fixed Domains

In DLs, models can be of arbitrary cardinality. In many applications, however, the domain of interest is known to be finite. In fact, restricting reasoning to models of finite domain size (called *finite model reasoning*, a natural assumption in database theory), has already become the focus of intense studies in DLs [17, 6, 25, 27]. As opposed to assuming the domain to be merely finite (but of arbitrary, unknown size), we consider the case where the domain has an *a priori known cardinality* and use the term *fixed domain*. We refer to such models as *fixed-domain models* and argue that in many applications, this modification of the standard DL semantics represents a more intuitive definition of what is considered and expected as a *model* of some knowledge base.

Definition 1 (Fixed-Domain Semantics) *Given a non-empty finite set $\Delta \subseteq N_I$, called* fixed domain*, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is said to be* Δ-fixed *(or just* fixed*, if Δ is clear from the context), if $\Delta^{\mathcal{I}} = \Delta$ and $a^{\mathcal{I}} = a$ for all $a \in \Delta$. Accordingly, for a DL knowledge base $\mathcal{K}$, we call an interpretation $\mathcal{I}$ a* Δ-model *of $\mathcal{K}$, 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. We say $\mathcal{K}$* Δ-entails *an axiom α ($\mathcal{K} \models_\Delta \alpha$) if every Δ-model of $\mathcal{K}$ is also a model of α.*

We briefly demonstrate the effects of the fixed-domain semantics as opposed to the finite-model semantics (with entailment $\models_{\text{fin}}$) and the classical semantics.

Example 2 *Let $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ and $\Delta = \{a, b\}$ with $\mathcal{A} = \{A(a), A(b), s(a, b)\}$, $\mathcal{T} = \{\top \sqsubseteq \exists r.B, \top \sqsubseteq \leqslant 1\, r^-.\top\}$, and $\mathcal{R} = \{\mathsf{Dis}(s, r)\}$. First we note that $\mathcal{K}$ has a Δ-model $\mathcal{I}$ representable as $\mathcal{A}_{\mathcal{I}} = \{A(a), A(b), B(a), B(b), s(a, b), r(a, a), r(b, b)\}$, thus $\mathcal{K}$ is satisfiable under all three semantics. Then $\alpha = \top \sqsubseteq \exists r.\exists r.B$ holds in all models of $\mathcal{K}$, therefore $\mathcal{K} \models \alpha$, $\mathcal{K} \models_{\text{fin}} \alpha$, and $\mathcal{K} \models_\Delta \alpha$. Opposed to this, $\beta = \top \sqsubseteq B$ merely holds in all finite models, whence $\mathcal{K} \models_{\text{fin}} \beta$ and $\mathcal{K} \models_\Delta \beta$, but $\mathcal{K} \not\models \beta$. Finally, $\gamma = \top \sqsubseteq \exists r.Self$ only holds in all Δ-models, thus $\mathcal{K} \models_\Delta \gamma$, but $\mathcal{K} \not\models_{\text{fin}} \gamma$ and $\mathcal{K} \not\models \gamma$.*

Extraction & Enumeration of Δ-Models When performing knowledge base satisfiability checking in DLs (the primary reasoning task usually considered), a model constructed by a reasoner merely serves as witness to claim satisfiability, rather than as an accessible artifact. However, as mentioned before, our approach aims at scenarios where a knowledge base is a formal problem description for which each model represents one solution; in particular the domain is part of the problem description and hence fixed a-priori. Then, retrieval of one, several, or all models is a natural task, as opposed to merely checking model existence. With *model extraction* we denote the task of materializing an identified model in order to be able to work with it, i.e. to inspect it in full detail and reuse it in downstream processes. The natural extension of model extraction is to make *all* models explicit, performing *model enumeration*. Most existing DL reasoning algorithms attempt to successively construct a model representation of a given knowledge base. However, most of the existing tableaux reasoners do not reveal the constructed model, besides the fact that models might end up being infinite such that an explicit representation is impossible. Regarding enumeration, we state that this task is not supported – not even implicitly – by any state-of-the-art DL reasoner, also due to the reason that in the standard case, the number of models is typically infinite and often even uncountable. We will use the notions of model extraction and enumeration in the way described above. Note that a related task, called *model expansion*, is used in the general first-order case, e.g. in the work of Mitchell and Ternovska [18]. There, an initial (partial) interpretation representing a problem instance is expanded to ultimately find a model of the given theory.

Example 3 *We consider the 3-coloring problem for an undirected graph $G = (V, E)$, encoded in $\mathcal{K}_{3\text{col}} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$, with $\mathcal{T} = \{N \sqsubseteq N_r \sqcup N_g \sqcup N_b,\ N_r \sqsubseteq \forall edge.(N_g \sqcup N_b),\ N_g \sqsubseteq \forall edge.(N_b \sqcup N_r),\ N_b \sqsubseteq \forall edge.(N_r \sqcup N_g), N_r \sqsubseteq \neg N_g, N_r \sqsubseteq \neg N_b, N_g \sqsubseteq \neg N_b\}$, $\mathcal{A} = \{N(v) \mid v \in V\} \cup \{edge(v, v') \mid \forall\{v, v'\} \in E\} \cup \{\neg edge(v, v') \mid \{v, v'\} \in 2^V \setminus E\}$, and $\mathcal{R} = \emptyset$. Let $\Delta = V$ be the imposed fixed domain. It is not hard to see that there is a one-to-one correspondence between the Δ-models of $\mathcal{K}_{3\text{col}}$ and the colorings of G.*

Axiomatization of Δ-Models When introducing a new semantics for some logic, it is worthwhile to ask if existing reasoners can be used. Indeed, it is easy to see that, assuming $\Delta = \{a_1, \ldots, a_n\}$, adding the GCI $\top \sqsubseteq \{a_1, \ldots, a_n\}$ as well as the set of inequality axioms containing $a_i \neq a_j$ with $i < j$ to $\mathcal{K}$ will (up to isomorphism) rule out all models of $\mathcal{K}$, not having Δ as their domain. Denoting these additional axioms with $\mathcal{FD}_\Delta$, we find that $\mathcal{K}$ is Δ-satisfiable iff $\mathcal{K} \cup \mathcal{FD}_\Delta$ is satisfiable under the classical DL semantics and, likewise, $\mathcal{K} \models_\Delta \alpha$ iff $\mathcal{K} \cup \mathcal{FD}_\Delta \models \alpha$ for any axiom α. Consequently, any off-the-shelf $\mathcal{SROIQ}$ reasoner can be used for fixed-domain reasoning, at least when it comes to the classical reasoning tasks.

However, the fact that the currently available DL reasoners are not optimized towards reasoning with axioms of the prescribed type (featuring disjunctions over potentially large sets of individuals) and that available reasoners do not support model extraction and model enumeration led us to develop an alternative computational approach based on ASP.

4 Complexity Analysis

In this section we investigate complexities of classical reasoning tasks under the fixed-domain semantics. Note that, next

to the size of the considered knowledge base (and - in the case of query entailment - the size of the query) the size of the domain $|\Delta|$ contributes to the input size of the reasoning problems considered.

Standard Reasoning The combined complexity of standard reasoning in $\mathcal{SROIQ}$ is known to be N2ExpTime-complete, both for arbitrary models and finite models [15]. Restricting to fixed domains leads to a drastic drop in complexity. Contrarily, imposing fixed domains on (allegedly) inexpressive fragments such as DL-Lite$_{\mathsf{core}}$, turns reasoning into a harder problem.

Let $\mathsf{DL}_{\mathsf{min}}$ be a minimalistic description logic that merely allows TBox axioms of the form $A \sqsubseteq \neg B$, with $A, B \in \mathsf{N}_C$. Moreover, only atomic assertions of the form $A(a)$ and $r(a,b)$ are admitted. We first demonstrate that satisfiability checking in $\mathsf{DL}_{\mathsf{min}}$ is NP-hard, allowing us to bequeath hardness up to more expressive DLs such as $\mathcal{SROIQ}$. Subsequently, we demonstrate that fixed-domain satisfiability checking in $\mathcal{SROIQ}$ is in NP, thus obtaining NP-completeness for all languages between $\mathsf{DL}_{\mathsf{min}}$ and $\mathcal{SROIQ}$.

Lemma 4 *The combined complexity of checking fixed-domain satisfiability of a* $\mathsf{DL}_{\mathsf{min}}$ *knowledge base* $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ *is* NP*-hard.*

Proof (Sketch) We obtain hardness by a reduction of the 3-colorability problem. Let $G = (V, E)$ be the input graph. Then, for each node $v_i \in V = \{v_1, \ldots, v_n\}$ we introduce a concept name V_i, and encode the edges as disjointness axioms, such that $\mathcal{T} = \{V_i \sqsubseteq \neg V_j \mid \{v_i, v_j\} \in E\}$. The ABox $\mathcal{A}$ consists of the assertions $V_i(a_i)$ for each $V_i \in \{V_1, \ldots, V_n\}$. Now let $\Delta = \{r, g, b\}$, such that under any Δ-fixed interpretation $\mathcal{I}$, necessarily $a_i^{\mathcal{I}} \in \{r, g, b\}$, $1 \leq i \leq n$. Consequently, G has a 3-coloring, iff $\mathcal{K} = (\mathcal{A}, \mathcal{T})$ is Δ-satisfiable. The reduction is linear in the size of G. □

Lemma 5 *The combined complexity of checking fixed-domain satisfiability of* $\mathcal{SROIQ}$ *knowledge bases is in* NP*.*

Proof (Sketch) Let $\mathcal{K}$ be a $\mathcal{SROIQ}$ knowledge base and Δ be the fixed domain. To show membership, we note that after guessing a Δ-fixed interpretation $\mathcal{I}$, modelhood can be checked in polynomial time. For this we let $\mathcal{C}$ contain all the concept expressions occurring in $\mathcal{K}$ (including subexpressions). Furthermore, let $\mathcal{R}$ contain all role expressions and role chains (including subchains) occurring in $\mathcal{K}$. Obviously, $\mathcal{C}$ and $\mathcal{R}$ are of polynomial size. Then, in a bottom-up fashion, we can compute the extension $C^{\mathcal{I}}$ of every element C of $\mathcal{C}$ and the extension $r^{\mathcal{I}}$ of every element r of $\mathcal{R}$ along the defined semantics. Obviously, each such computation step requires only polynomial time. Finally, based on the computed extensions, every axiom of $\mathcal{K}$ can be checked – again in polynomial time. □

Combining these propositions yields the following theorem.

Theorem 6 *Fixed-domain satisfiability checking in any language between* $\mathsf{DL}_{\mathsf{min}}$ *and* $\mathcal{SROIQ}$ *is* NP*-complete.*

Note that this finding contrasts with the observation that fixed-domain reasoning in first-order logic is PSpace-complete. We omit the full proof here, just noting that membership and hardness can be easily shown based on the fact that checking modelhood in FOL is known to be PSpace-complete [31] and, for the membership part, keeping in mind that NPSpace = PSpace thanks to Savitch's Theorem [28]. This emphasizes the fact that, while the fixed-domain restriction turns reasoning in FOL decidable, restricting to $\mathcal{SROIQ}$ still gives a further advantage in terms of complexity (assuming NP ≠ PSpace).

Query Entailment We next consider the complexity of query entailment for DLs. Again, we will notice a very uniform behavior over a wide range of DLs and query types. We will start by showing a hardness result for a very minimalistic setting.

Lemma 7 *The combined complexity of fixed-domain entailment of conjunctive queries from a* $\mathsf{DL}_{\mathsf{min}}$ *knowledge base is* Π_2^{p}*-hard.*

Proof We show hardness by providing a polynomial reduction from evaluation of quantified Boolean formulae of the form $\Phi = \forall p_1, \ldots, p_\ell \exists q_1, \ldots, q_m \varphi$ such that φ is a Boolean formula where the propositional symbols are from the set $\{p_1, \ldots, p_\ell, q_1, \ldots, q_m\}$. Note that w.l.o.g. we can assume φ to be in conjunctive normal form, i.e. it has the shape $\bigvee L_1 \wedge \ldots \wedge \bigvee L_n$ where the L_i are sets of negated or unnegated propositional symbols.

Given such a formula Φ, we now construct a $\mathsf{DL}_{\mathsf{min}}$ knowledge base $\mathcal{K}$, a domain Δ, and a conjunctive query Q (all of polynomial size) such that $\mathcal{K}$ $\Delta_{\mathcal{K}}$-entails Q if and only if Φ evaluates to true. We let Δ consist of elements d_t^{true} and d_t^{false} for all $t \in \{p_1, \ldots, p_\ell, q_1, \ldots, q_m\}$, and $\mathcal{K}$ consist of the axioms:

- $\mathit{InClause}_L(d_t^{\text{true}})$ whenever $t \in L$ and $\mathit{InClause}_L(d_t^{\text{false}})$ whenever $\neg t \in L$
- $\mathit{compatible}(d_t^{\text{true}}, d_u^{\text{true}})$ and $\mathit{compatible}(d_t^{\text{false}}, d_u^{\text{false}})$ for all $\{t, u\} \subseteq \{p_1, \ldots, p_\ell, q_1, \ldots, q_m\}$
- $\mathit{compatible}(d_t^{\text{false}}, d_u^{\text{true}})$ and $\mathit{compatible}(d_t^{\text{true}}, d_u^{\text{false}})$ for all $\{t, u\} \subseteq \{p_1, \ldots, p_\ell, q_1, \ldots, q_m\}$ with $t \neq u$
- $\mathit{Select}(d_t)$, $C_t(d_t)$ for all $t \in \{p_1, \ldots, p_\ell\}$
- $\mathit{Select}(d_t^{\text{true}})$ and $\mathit{Select}(d_t^{\text{false}})$ for all $t \in \{q_1, \ldots, q_m\}$
- $C_t(d_t^{\text{true}})$, $C_t(d_t^{\text{false}})$ for all $t \in \{p_1, \ldots, p_\ell, q_1, \ldots, q_m\}$
- $C_t \sqcap C_u \sqsubseteq \bot$ for all $\{t, u\} \in \{p_1, \ldots, p_\ell, q_1, \ldots, q_m\}$ with $t \neq u$

Finally, we let Q be the conjunctive query using the variables $x_{L_1}, \ldots, x_{L_n}$ and consisting of the atoms $\mathit{InClause}_L(x_L)$, $\mathit{Select}(x_L)$ for all $L \in \{L_1, \ldots, L_n\}$ as well as $\mathit{compatible}(x_L, x_{L'})$ for all $\{L, L'\} \in \{L_1, \ldots, L_n\}$.

We now sketch the argument why the above claimed correspondence holds. By construction, the minimal $\Delta_{\mathcal{K}}$-models $\mathcal{I}$ for every $i \in \{1, \ldots, m\}$ are exactly those where (next to the explicitly stated concept and role memberships) either $d_{p_i}^{\text{true}} \in \mathit{Select}^{\mathcal{I}}$ or $d_{p_i}^{\text{false}} \in \mathit{Select}^{\mathcal{I}}$ holds. Consequently Q is entailed, iff for each of these models (representing all possible truth assignments to $p_1, \ldots, p_\ell$), one literal from every clause

L_i can be selected such that (a) this selection is consistent (i.e., no contradicting literals are selected) and (b) whenever a literal w.r.t. $p_1, \ldots, p_\ell$ is selected, it must be the one corresponding with the model's predefined truth assignment for these propositional symbols. However, this is the case exactly if Φ is valid. □

We continue by showing that even for very expressive DLs and query languages, query entailment under the fixed domain semantics is still in the second level of the polynomial hierarchy.

Lemma 8 *The combined complexity of the fixed-domain entailment of bounded-arity Datalog queries from a $\mathcal{SROIQ}$ knowledge base is in Π_2^{P}.*

Proof Satisfaction of a bounded-arity Datalog query in a database (or finite interpretation) is in NP: there are only polynomially many ground atoms that can be derived, hence, whenever the query is entailed, there is a ground proof tree of polynomial size which can be verified in polynomial time. Consequently, fixed-domain non-entailment of such a query Q from a $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ can be realized by (a) guessing an interpretation $\mathcal{I}$ (b) verifying $\mathcal{I} \models \mathcal{K}$ in polynomial time (cf. the proof of Lemma 5) and (c) using an NP oracle to verify $\mathcal{I} \not\models Q$. Consequently, checking fixed-domain entailment is in $\mathrm{coNP}^{\mathrm{NP}} = \Pi_2^{\mathrm{P}}$. □

Bounded-arity Datalog queries over DLs are rather expressive, they subsume many of the prominent query classes in knowledge representation and databases, including (unions of) conjunctive queries, positive queries, (unions of) conjunctive 2-way regular path queries [7], positive 2-way regular path queries, (unions of) conjunctive nested 2-way regular path queries [3] and regular queries as defined in [24]. Combining the two propositions, we obtain the following theorem.

Theorem 9 *For any class of queries subsuming conjunctive queries and subsumed by bounded-arity Datalog queries and any DL subsuming $\mathsf{DL}_{\mathsf{min}}$ and subsumed by $\mathcal{SROIQ}$, the combined complexity of fixed-domain query entailment is Π_2^{P}-complete.*

5 Practical Fixed-Domain Reasoning

In Section 3 we already claimed that available reasoners perform poorly on knowledge bases when axiomatizing the fixed-domain semantics, and we support this statement with an evaluation in the sequel (cf. Section 5.2). Thus, a more viable approach is required when considering practical reasoning. To this end, we propose an encoding of arbitrary $\mathcal{SROIQ}$ knowledge bases into *answer set programs*. This allows us to use existing ASP machinery to perform both standard reasoning as well as the non-standard tasks *model extraction* & *enumeration* and query entailment quite elegantly.

5.1 ASP Encoding of DL Knowledge Bases

We now describe how standard and non-standard reasoning tasks w.r.t. the fixed-domain semantics can be encoded by ASP. Intuitively, the set of all Δ-interpretations defines a search space, which can be traversed searching for Δ-models, guided by appropriate constraints. We thus propose a translation $\Pi(\mathcal{K}, \Delta)$ for any $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$; i.e. $\Pi(\mathcal{K}, \Delta) = \Pi_{\mathrm{gen}}(\Delta) \cup \Pi_{\mathrm{chk}}(\mathcal{K})$, consisting of a generating part $\Pi_{\mathrm{gen}}(\Delta)$ that defines all potential candidate interpretations, and a constraining part $\Pi_{\mathrm{chk}}(\mathcal{K})$ that rules out interpretations violating axioms in $\mathcal{K}$.

Simplified Form of Knowledge Bases We first impose a few additional assumptions regarding the knowledge base $\mathcal{K}$. To start with, we assume that the ABox only refers to domain individuals from Δ and does not mention elements from $N_I(\mathcal{K}) \setminus \Delta$ (except in nominal concepts). This can be obtained by the following standard model-preserving transformations (let $a, b \in N_I(\mathcal{K}) \setminus \Delta$ while $c, d \in \Delta$):

$$\begin{array}{lcl@{\qquad}lcl} C(a) & \rightsquigarrow & \{a\} \sqsubseteq C & a \doteq b & \rightsquigarrow & \{a\} \sqsubseteq \{b\} \\ r(a,b) & \rightsquigarrow & \{a\} \sqsubseteq \exists r.\{b\} & a \not\doteq b & \rightsquigarrow & \{a\} \sqsubseteq \neg\{b\} \\ r(c,b) & \rightsquigarrow & \exists r.\{b\}(c) & c \doteq b & \rightsquigarrow & \{b\}(c) \\ r(a,d) & \rightsquigarrow & \exists r^-.\{a\}(d) & c \not\doteq b & \rightsquigarrow & \neg\{b\}(c) \end{array}$$

In the light of this, it is also clear that we can assume the ABox to be free of (in)equality statements: between individuals from Δ they are either tautological (and can be removed) or obviously contradictory (and could be replaced by $\top \sqsubseteq \bot$).

Next, we make the common assumption that all concept expressions and all roles occurring in the ABox are concept and role names, respectively (which is easy to establish by the use of auxiliary concepts). Further, we impose the assumption that $\mathcal{K}$ does not contain any nominal concepts. This can be realized by replacing every occurrence of a nominal concept $\{a\}$ in $\mathcal{K}$ with a fresh auxiliary atomic concept $O_{\{a\}}$, adding the axiom $\mathsf{Sing}(O_{\{a\}})$ (which we use as a shortcut for the two axioms $\top \sqsubseteq \leqslant 1\, u.O_{\{a\}}$ and $\top \sqsubseteq \geqslant 1\, u.O_{\{a\}}$ in order to enforce that $O_{\{a\}}$ must be interpreted by a singleton set) to the TBox and, in case $a \in \Delta$, adding the assertion $O_{\{a\}}(a)$ to the ABox. Obviously, there is a one-to-one correspondence between (Δ-)models $\mathcal{I}$ of the original and (Δ-)models $\mathcal{J}$ of the transformed knowledge base via $a^{\mathcal{I}} = \delta$ iff $O_{\{a\}}^{\mathcal{J}} = \{\delta\}$, hence ($\Delta$-)models of the original knowledge base are straightforward to reconstruct.

Our last requirement is that the knowledge base is in normalized form, obtained by a modified structural transformation $\Omega(\mathcal{K})$, akin to the one proposed in [21]. A TBox axiom is normalized, if it is of the form $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$, where C_i is of the form B, $\forall r.B$, $\exists r.\mathit{Self}$, $\neg\exists r.\mathit{Self}$, $\geq n\, r.B$, or $\leq n\, r.B$, for B a literal concept (i.e., a concept name or a negated concept name), r a role, and n a positive integer. It can be shown that the obtained normalized knowledge base $\Omega(\mathcal{K})$ is a model-conservative extension of $\mathcal{K}$, i.e. every (Δ-)model of $\Omega(\mathcal{K})$ is a (Δ-)model of $\mathcal{K}$ and every (Δ-)model of $\mathcal{K}$ can be turned into a (Δ-)model of $\Omega(\mathcal{K})$ by finding appropriate interpretations for the concepts and roles introduced by Ω. Thereby it is straightforward to extract a model for $\mathcal{K}$, given a model of $\Omega(\mathcal{K})$. Table 3 depicts the normalization rules in detail. Thanks to the correspondences depicted here, we can for the rest of our considerations assume that the considered knowledge base is free of mentions of non-Δ individuals, free of nominals, free of (in)equality statements, and normalized; we call such a knowledge base *simplified*.

Table 3. Ω-Normalization of knowledge base axioms.

$$
\begin{array}{rcl}
\Omega(\mathcal{K}) & = & \bigcup_{\alpha\in\mathcal{R}\cup\mathcal{A}} \Omega(\alpha) \cup \bigcup_{C_1\sqsubseteq C_2\in\mathcal{T}} \Omega(\top \sqsubseteq \mathsf{nnf}(\neg C_1 \sqcup C_2)) \\
\Omega(\top \sqsubseteq \mathbf{C} \sqcup C') & = & \Omega(\top \sqsubseteq \mathbf{C} \sqcup \alpha_{C'}) \cup \bigcup_{1\le i\le n} \Omega(\top \sqsubseteq \dot{\neg}\alpha_{C'} \sqcup C_i) \\
 & & \text{for } C' \text{ of the form } C' = C_1 \sqcap \cdots \sqcap C_n \text{ and } n \ge 2 \\
\Omega(\top \sqsubseteq \mathbf{C} \sqcup \exists r.D) & = & \Omega(\top \sqsubseteq \mathbf{C} \sqcup \geq 1\ r.D) \\
\Omega(\top \sqsubseteq \mathbf{C} \sqcup \forall r.D) & = & \Omega(\top \sqsubseteq \mathbf{C} \sqcup \forall r.\alpha_D) \cup \Omega(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup D) \\
\Omega(\top \sqsubseteq \mathbf{C} \sqcup \geq n\ r.D) & = & \Omega(\top \sqsubseteq \mathbf{C} \sqcup \geq n\ r.\alpha_D) \cup \Omega(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup D) \\
\Omega(\top \sqsubseteq \mathbf{C} \sqcup \leq n\ r.D) & = & \Omega(\top \sqsubseteq \mathbf{C} \sqcup \leq n\ r.\dot{\neg}\alpha_{\dot{\neg}D}) \cup \Omega(\top \sqsubseteq \dot{\neg}\alpha_{\dot{\neg}D} \sqcup \dot{\neg}D) \\
\Omega(D(s)) & = & \{\alpha_D(s)\} \cup \Omega(\top \sqsubseteq \dot{\neg}\alpha_D \sqcup \mathsf{nnf}(D)) \\
\Omega(r^-(s,t)) & = & \{r(t,s)\} \\
\Omega(r_1 \circ \ldots \circ r_n \sqsubseteq r) & = & \{r_1 \circ r_2 \sqsubseteq r_{(r_1 \circ r_2)}\} \cup \Omega(r_{(r_1\circ r_2)} \circ r_3 \circ \ldots \circ r_n \sqsubseteq r) \\
 & & \text{for any RIA with } n > 2 \\
\Omega(\beta) & = & \{\beta\} \text{ for any other axiom } \beta
\end{array}
$$

$$
\alpha_C = \begin{cases} Q_C & \text{if } \mathsf{pos}(C) = \mathit{true} \\ \neg Q_C & \text{if } \mathsf{pos}(C) = \mathit{false} \end{cases}, \text{where } Q_C \text{ is a fresh concept name unique for } C.
$$

$$
\begin{array}{rcl@{\qquad}rcl}
\mathsf{pos}(\top) & = & \mathit{false} & \mathsf{pos}(\bot) & = & \mathit{false} \\
\mathsf{pos}(A) & = & \mathit{true} & \mathsf{pos}(\neg A) & = & \mathit{false} \\
\mathsf{pos}(\exists r.Self) & = & \mathit{true} & \mathsf{pos}(\neg\exists r.Self) & = & \mathit{false} \\
\mathsf{pos}(C_1 \sqcap C_2) & = & \mathsf{pos}(C_1) \vee \mathsf{pos}(C_2) & \mathsf{pos}(C_1 \sqcup C_2) & = & \mathsf{pos}(C_1) \vee \mathsf{pos}(C_2) \\
\mathsf{pos}(\forall r.C_1) & = & \mathsf{pos}(C_1) & \mathsf{pos}(\leq n\ r.C_1) & = & \begin{cases} \mathsf{pos}(\dot{\neg}C_1) & \text{if } n = 0 \\ \mathit{true} & \text{otherwise} \end{cases} \\
\mathsf{pos}(\geq n\ r.C_1) & = & \mathit{true} & & &
\end{array}
$$

Note: A is an atomic concept, $C_{(i)}$ are arbitrary concept expressions, $\mathbf{C}$ is a possibly empty disjunction of concept expressions, D is **not** a literal concept. The function $\dot{\neg}$ is defined as $\dot{\neg}(\neg A) = A$ and $\dot{\neg}(A) = \neg A$ for some atomic concept A.

Candidate Generation Following the generate & test paradigm, we let $\Pi_{\text{gen}}(\Delta)$ be the program that generates (all) possible interpretations over Δ; i.e. for each concept name A and role name r all possible extensions over Δ are generated (while the interpretation of the individual names is just the identity, exploiting our assumption that $\mathcal{K}$ does not contain non-Δ individual names). For each concept name A and role name r occurring in $\mathcal{K}$, $\Pi_{\text{gen}}(\Delta)$ contains the following rules:

$$A(X) \leftarrow \top(X), \mathit{not}\ \bar{A}(X). \quad (1)$$

$$\bar{A}(X) \leftarrow \top(X), \mathit{not}\ A(X). \quad (2)$$

$$r(X,Y) \leftarrow \top(X), \top(Y), \mathit{not}\ \bar{r}(X,Y). \quad (3)$$

$$\bar{r}(X,Y) \leftarrow \top(X), \top(Y), \mathit{not}\ r(X,Y). \quad (4)$$

Thereby, slightly overloading notation, A and $\bar{A}$ are unary predicates introduced for every concept name A in $\mathcal{K}$ and r and $\bar{r}$ are binary predicates introduced for every role name r. Moreover, we use $\top$ as unary domain predicate and let $\Pi_{\text{gen}}(\Delta)$ contain the fact $\top(\delta)$ for each domain element $\delta \in \Delta$. Rules (1) and (2) guess a set of ground instances of $A(X)$ for any concept name A, while Rules (3) and (4) realize the same for role names r. Further, we have to axiomatize the universal role u as follows:

$$u(X,Y) \leftarrow \top(X), \top(Y). \quad (5)$$

Like this, an answer set $\mathbf{A}$ of $\Pi_{\text{gen}}(\Delta)$ directly corresponds to an interpretation $\mathcal{I}_{\mathbf{A}} = (\Delta, \cdot^{\mathcal{I}_{\mathbf{A}}})$ of $\mathcal{K}$ over the fixed-domain Δ as follows:

$$A^{\mathcal{I}_{\mathbf{A}}} = \{\delta \mid A(\delta) \in \mathbf{A}\}, \text{ for all } A \in N_C(\mathcal{K}),$$
$$r^{\mathcal{I}_{\mathbf{A}}} = \{(\delta,\delta') \mid r(\delta,\delta') \in \mathbf{A}\}, \text{ for all } r \in N_R(\mathcal{K}),$$
$$a^{\mathcal{I}_{\mathbf{A}}} = a.$$

Axiom Encoding Beginning with assertions in $\mathcal{A}$, for each $A(a)$ and each role $r(a,b)$ we add the identical fact to $\Pi_{\text{chk}}(\mathcal{K})$.

Further, we turn each axiom $\alpha \in \mathcal{T} \cup \mathcal{R}$ into a constraint, ultimately ruling out those candidate interpretations not satisfying α, whence $\Pi_{\text{chk}}(\mathcal{K}) = \mathcal{A} \cup \Pi_{\text{chk}}(\mathcal{T}) \cup \Pi_{\text{chk}}(\mathcal{R})$. Since each $\alpha \in \mathcal{T}$ is of the form $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$, we simply turn it into a negative constraint of the form $\prod_{i=1}^{n} \neg C_i \sqsubseteq \bot$, and add its direct translation to $\Pi_{\text{chk}}(\mathcal{T})$. Thus, for each $\top \sqsubseteq \bigsqcup_{i=1}^{n} C_i$ a constraint of the following form is obtained, where the individual translation $\mathit{trans}(C_i)$ of concepts C_i is defined in Table 4:

$$\leftarrow \mathit{trans}(C_1), \ldots, \mathit{trans}(C_n). \quad (6)$$

Role assertions and role inclusion axioms are also turned into constraints, and we add their direct translation to $\Pi_{\text{chk}}(\mathcal{R})$. For each role inclusion $r \sqsubseteq s \in \mathcal{R}$, role disjointness $\mathsf{Dis}(r,s) \in \mathcal{R}$ and role chain $s_1 \circ s_2 \sqsubseteq r \in \mathcal{R}$ this yields

$$\leftarrow r(X,Y),\ \mathit{not}\ s(X,Y). \quad (7)$$

$$\leftarrow s(X,Y),\ r(X,Y). \quad (8)$$

$$\leftarrow s_1(X,Y), s_2(Y,Z),\ \mathit{not}\ r(X,Z). \quad (9)$$

respectively, where we silently assume that an expressions of the form $r^-(V_1,V_2)$ represents the atom $r(V_2,V_1)$.

Theorem 10 *Let $\mathcal{K}$ be a simplified $\mathcal{SROIQ}$ knowledge base. Then, $\mathcal{I} \models_\Delta \mathcal{K}$ exactly if $\{A(\delta) \mid \delta \in A^\mathcal{I}, A \in N_C(\mathcal{K}) \cup \{\top\}\} \cup \{r(\delta, \delta') \mid (\delta, \delta') \in r^\mathcal{I}, r \in N_R(\mathcal{K}) \cup \{u\}\}$ is an answer set of $\Pi(\mathcal{K}, \Delta)$.*

This theorem establishes a tight one-to-one correspondence between Δ-models of a knowledge base and answer sets of its ASP translation, and allows us to employ ASP solving tools for fixed-domain DL reasoning. Most notably, in addition to the standard DL reasoning tasks, *model extraction* and *model enumeration* can be carried out without additional efforts, since both are natural tasks for answer set solvers. Moreover, all mentioned query formalisms can be straightforwardly expressed in a rule-based way, whence integration in our framework is immediate.

Table 4. Translation of Concept Expressions.

C	$trans(C)$
A	$not\ A(X)$
$\neg A$	$A(X)$
$\forall r.A$	$\{not\ A(Y_A),\ r(X, Y_A)\}$
$\forall r.\neg A$	$\{r(X, Y_A),\ A(Y_A)\}$
$\exists r.Self$	$not\ r(X, X)$
$\neg\exists r.Self$	$r(X, X)$
$\geq n\ r.A$	$\#count\{r(X, Y_A)\ :\ A(Y_A)\} < n$
$\geq n\ r.\neg A$	$\#count\{r(X, Y_A)\ :\ not\ A(Y_A)\} < n$
$\leq n\ r.A$	$\#count\{r(X, Y_A)\ :\ A(Y_A)\} > n$
$\leq n\ r.\neg A$	$\#count\{r(X, Y_A)\ :\ not\ A(Y_A)\} > n$

5.2 Implementation and Experiments

We implemented our translation based approach as an open-source tool – named `Wolpertinger`.[3] The obtained logic programs can be evaluated with most modern ASP solvers. However, the evaluation was conducted using `Clingo` [9] for grounding and solving, since it is currently the most prominent solver leading the latest competitions [5].

In fact, the tool is used for solving nontrivial real-world configuration problems in an industry project with very good results. Yet, for legal reasons we cannot disclose any details.

Hence, we present preliminary evaluation results based on simple ontologies, encoding constraint-satisfaction-type combinatorial problems. Existing OWL ontologies typically used for benchmarking, e.g. SNOMED or GALEN [29, 23], do not fit our purpose, since they are modeled with the classical semantics in mind and often have little or no ABox information.

Our tests provide runtimes compared to the popular `HermiT` and `Konclude` reasoners [12, 30]. Both are full OWL 2 DL reasoners and are leading the latest competitions. Whereas a direct comparison would not be fair, the conducted tests shall merely show the feasibility of our approach in comparison to standard DL reasoners using the axiomatization. In particular we focus on model enumeration, for which we can not conduct any comparison with existing DL reasoners. The evaluation itself is conducted on a standard desktop machine (Unix operating system, 2.7 Ghz Intel Core i5 Processor, 8 GB memory and standard Java-VM settings).

[3] `https://github.com/wolpertinger-reasoner/Wolpertinger`

Unsatisfiability We construct an unsatisfiable knowledge base $\mathcal{K}_n = (\mathcal{A}_n, \mathcal{T}_n, \emptyset)$, with $\mathcal{T}_n = \{A_1 \sqsubseteq \exists r.A_2, \ldots, A_n \sqsubseteq \exists r.A_{n+1}, A_i \sqcap A_j \sqsubseteq \bot \mid 1 \leq i < j \leq n+1\}$ and $\mathcal{A}_n = \{A_1(a_1), \top(a_1), \ldots, \top(a_n)\}$, together with the fixed-domain $\Delta = \{a_1, \ldots, a_n\}$.

Inspired by common pigeonhole-type problems, we have $\mathcal{K}_n$ enforce an r-chain of length $n+1$ without repeating elements, yet, having fixed Δ to n elements such a model cannot exist. Table 5 depicts the runtimes for detecting unsatisfiability of $\mathcal{K}_n$, for increasing n. The durations correspond to the pure solving time as stated by the tools (including grounding in the case of `Clingo`), and neglecting pre-processing time. As the figures suggest, $\mathcal{K}_n$ is a potential worst-case scenario, where any of the tools is doomed to test all combinations. Whereas `Wolpertinger` is faster in claiming inconsistency in all cases up to $\mathcal{K}_{10}$, `HermiT` is slightly faster up from $\mathcal{K}_{11^b}$ – both leaving `Konclude` behind. As $\mathcal{K}_{11}$ causes already a massive increase, in the runs 7–10 we restricted the search space for $\mathcal{K}_{11}$ by explicitly adding negative r-edges and thus allowing for a more fine grained evaluation. However, $\mathcal{K}_{12}$ is already beyond a feasible time bound for all reasoners.

Table 5. Runtimes: Detecting Unsatisfiability of $\mathcal{K}_n$.

#	$\mathcal{K}_n$	`Wolpertinger`	`HermiT`	`Konclude`
1	5	< 0.01 s	0.48 s	0.04 s
2	6	< 0.01 s	0.67 s	0.07 s
3	7	0.04 s	0.94 s	0.26 s
4	8	0.33 s	1.81 s	1.79 s
5	9	3.72 s	9.52 s	16.19 s
6	10	68.53 s	87.88 s	152.37 s
7	11^a	111.43 s	203.67 s	342.90 s
8	11^b	350.13 s	301.15 s	516.60 s
9	11^c	438.70 s	403.69 s	669.47 s
10	11^d	582.90 s	491.34 s	878.46 s
11	11	1 095.49 s	1 027.33 s	1 682.41 s

Table 6. Runtimes: Sudoku Satisfiability.

#	Size	`Wolpertinger`	`HermiT`	`Konclude`
1	6 × 6 - 0	1.04 s	6.32 s	2.75 s
2	9 × 9 - 46	5.68 s	0.93 s	11.37 s
3	9 × 9 - 42	5.78 s	1.06 s	20.68 s
4	9 × 9 - 38	6.96 s	1.47 s	60.17 s
5	9 × 9 - 28	6.87 s	27.48 s	> 30 min
6	9 × 9 - 0	6.73 s	> 30 min	> 30 min

Table 7. Runtimes: Enumerating 9×9-Sudoku Instances.

#	Models	Time(Total)	Time(Solving)
1	100	6.73 s	0.11 s
2	1 000	7.16 s	0.33 s
3	10 000	9.06 s	2.39 s
4	100 000	29.27 s	22.53 s
5	1 000 000	225.40 s	218.56 s

Model Extraction and Enumeration With Table 6, we next provide some figures for model extraction and partial enumeration (retrieving a given number of Δ-models). To this

end, we created a knowledge base modeling fully and correctly filled Sudokus, beginning with a 6×6-instance, consisting of 64 individuals, 13 concept names and 1 role name. Then testing on a 9×9-instance featuring 108 named individuals. Whereas `HermiT` & `Konclude` are still able to claim satisfiability for the 6×6-instance, invoking a satisfiability test on the 9×9-instance (axiomatized knowledge base), no answer was given within 30 minutes. In both cases `Wolpertinger` detects satisfiability with reasonable runtimes. Therefore, we again restricted the search space in run 2-5 by having 46 cells pre-filled, respectively 42, 38 and 28.

For model enumeration, we used the knowledge base for the 9×9-instance and turning the task into generating new Sudoku instances. Table 7 depicts the figures, where it can be observed that, besides a constant time of around 6 seconds required for grounding, requesting 10^6 models is reasonably efficient.

6 Conclusion and Future Work

For OWL ontologies which represent constraint-type problems, the fixed-domain semantics allows to confine modelhood of interpretations to models of the right form. Although OWL still imposes some restrictions regarding expressivity (e.g., by restricting the arity of the used predicates to 1 and 2), we argue that quite large and involved problem scenarios can be modeled by OWL ontologies. Clearly, more comprehensive evaluations of our system with respect to such ontologies remain as imperative issue. Moreover, translations of fixed-domain reasoning problems into other formalisms are conceivable, including pure CSP languages or even SAT, which would have to be implemented and compared against the ASP approach.

Regarding our translation from DL to ASP, we want to point out that there is related work on computing finite models for general FO theories using ASP [10], where an incremental approach is implemented using iASP, in order to successively increase the size of the domain over which a model shall be constructed – ultimately proofing finite satisfiability. Although different in its motivation, it should be possible to compare both translations if we consider the FO translation of some DL knowledge base and the cardinality of the given fixed domain as initial domain size.

Another interesting strand of research would be to consider extensions of the source formalism, e.g. by non-monotonic features. As ASP itself is a non-monotonic logic programming formalism, rule-based extensions of OWL – monotonic [14, 20] or nonmonotonic [19, 1] – should be straightforward to accommodate. On yet another note, we plan to incorporate typical ontology engineering tasks such as explanation and axiom pinpointing into our ASP-based framework.

ACKNOWLEDGEMENTS

This work is supported by DFG in the Research Training Group QuantLA (GRK 1763). Moreover, we are grateful for all the valuable feedback from our colleagues and the anonymous reviewers, which helped greatly to improve this work.

REFERENCES

[1] José Júlio Alferes, Matthias Knorr, and Terrance Swift, 'Query-Driven Procedures for Hybrid MKNF Knowledge Bases', *ACM Transactions on Computational Logic*, **14**(2), 16:1–16:43, (2013).

[2] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, second edn., 2007.

[3] Meghyn Bienvenu, Diego Calvanese, Magdalena Ortiz, and Mantas Simkus, 'Nested regular path queries in description logics', in *Proc. of the 14th Conference on Principles of Knowledge Representation and Reasoning (KR 2014)*, eds., Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter. AAAI Press, (2014).

[4] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński, 'Answer set programming at a glance', *Commun. ACM*, **54**(12), 92–103, (2011).

[5] Francesco Calimeri, Martin Gebser, Marco Maratea, and Francesco Ricca, 'Design and results of the 5th answer set programming competition', *Artif. Intell.*, **231**, 151–181, (2016).

[6] Diego Calvanese, 'Finite model reasoning in description logics', in *Proc. of the International Workshop on Description Logics (DL 1996)*, eds., Lin Padgham, Enrico Franconi, Manfred Gehrke, Deborah L. McGuinness, and Peter F. Patel-Schneider, volume WS-96-05 of *AAAI Technical Report*, pp. 25–36. AAAI Press, (1996).

[7] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz, 'Regular path queries in expressive description logics with nominals', in *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, ed., Craig Boutilier, pp. 714–720, (2009).

[8] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub, 'Answer Set Solving in Practice', *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **6**, 1–238, (2012).

[9] Martin Gebser, Benjamin Kaufmann Roland Kaminski, Max Ostrowski Torsten Schaub, and Marius T. Schneider, 'Potassco: The Potsdam Answer Set Solving Collection', *AI Communications*, **24**(2), 107–124, (2011).

[10] Martin Gebser, Orkunt Sabuncu, and Torsten Schaub, 'Finite model computation via answer set programming', in *Proc. of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, ed., Toby Walsh, pp. 2626–2631. IJCAI/AAAI, (2011).

[11] Michael Gelfond and Vladimir Lifschitz, 'Classical negation in logic programs and disjunctive databases', *New Generation Comput.*, **9**(3/4), 365–386, (1991).

[12] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang, 'HermiT: an OWL 2 reasoner', *Journal of Automated Reasoning*, **53**(3), 245–269, (2014).

[13] Ian Horrocks, Oliver Kutz, and Ulrike Sattler, 'The Even More Irresistible $\mathcal{SROIQ}$', in *Proc. of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, eds., Patrick Doherty, John Mylopoulos, and Christopher A. Welty, pp. 57–67. AAAI Press, (2006).

[14] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin N. Grosof, and Mike Dean, *SWRL: A Semantic Web Rule Language*, W3C Member Submission, 21 May 2004.

[15] Yevgeny Kazakov, '$\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$', in *Proc. of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, eds., Gerhard Brewka and Jérôme Lang, pp. 274–284. AAAI Press, (2008).

[16] Holger Knublauch, Mark A. Musen, and Alan L. Rector, 'Editing Description Logic Ontologies with the Protégé OWL Plugin', in *Proc. of the International Workshop on Description Logics (DL 2004)*, eds., Volker Haarslev and Ralf Möller, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, (2004).

[17] Carsten Lutz, Ulrike Sattler, and Lidia Tendera, 'The com-

plexity of finite model reasoning in description logics', *Information and Computation*, **199**(1-2), 132–171, (May 2005).
[18] David G. Mitchell and Eugenia Ternovska, 'A framework for representing and solving NP search problems', in *Proc. of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference (AAAI 2005)*, eds., Manuela M. Veloso and Subbarao Kambhampati, pp. 430–435. AAAI Press / The MIT Press, (2005).
[19] Boris Motik and Riccardo Rosati, 'Reconciling description logics and rules', *Journal of the ACM*, **57**(5), (2010).
[20] Boris Motik, Ulrike Sattler, and Rudi Studer, 'Query Answering for OWL DL with Rules', *Journal of Web Semantics*, **3**(1), 41–60, (2005).
[21] Boris Motik, Rob Shearer, and Ian Horrocks, 'Hypertableau Reasoning for Description Logics', *Artificial Intelligence Research*, **36**, 165–228, (2009).
[22] Ilkka Niemelä, 'Logic programs with stable model semantics as a constraint programming paradigm', *Ann. Math. Artif. Intell.*, **25**(3-4), 241–273, (1999).
[23] Alan Rector and Ian Horrocks, 'Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions', in *Proc. of the Workshop on Ontological Engineering*, (1997).
[24] Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi, 'Regular Queries on Graph Databases', in *Proc. of the 18th International Conference on Database Theory (ICDT 2015)*, eds., Marcelo Arenas and Martín Ugarte, volume 31 of *LIPIcs*, pp. 177–194. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, (2015).
[25] Riccardo Rosati, 'Finite Model Reasoning in DL-Lite', in *Proc. of the 5th European Semantic Web Conference (ESWC 2008)*, eds., Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, volume 5021 of *LNCS*, p. 215. Springer, (2008).
[26] Sebastian Rudolph, 'Foundations of Description Logics', in *Reasoning Web. 7th International Summer School 2011, Tutorial Lectures*, eds., Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter F. Patel-Schneider, volume 6848 of *LNCS*, pp. 76–136. Springer, (2011).
[27] Sebastian Rudolph, 'Undecidability results for database-inspired reasoning problems in very expressive description logics', in *Proc. of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR 2016)*, (2016).
[28] Walter J. Savitch, 'Relationships Between Nondeterministic and Deterministic Tape Complexities', *J. Comput. Syst. Sci.*, **4**(2), 177–192, (April 1970).
[29] Kent A. Spackman, Keith E. Campbell, and Roger A. Côté, 'SNOMED RT: a reference terminology for health care', in *AMIA 1997, American Medical Informatics Association Annual Symposium*. AMIA, (1997).
[30] Andreas Steigmiller, Thorsten Liebig, and Birte Glimm, 'Konclude: System description', *Journal of Web Semantics*, **27**, 78–85, (2014).
[31] Larry J. Stockmeyer, *The Complexity of Decision Problems in Automata Theory and Logic*, Ph.D. dissertation, Massachusetts Institute of Technology, 1974.
[32] W3C OWL Working Group, *OWL 2 Web Ontology Language: Document Overview*, W3C Recommendation, 2009.