Reasoning with the Description Logic $\mathcal{DLRO}^{-\{\leq\}}$ using Bound Guarded Programs

Stijn Heymans
Digital Enterprise Research Institute (DERI)
University of Innsbruck, Austria

stijn.heymans@deri.org

Davy Van Nieuwenborgh[*]
Dept. of Computer Science
Vrije Universiteit Brussel, VUB
Pleinlaan 2, B1050 Brussels, Belgium

dvnieuwe@vub.ac.be

Dieter Fensel
Digital Enterprise Research Institute (DERI)
University of Innsbruck, Austria
National University of Ireland, Galway

dieter.fensel@deri.org

Dirk Vermeir
Dept. of Computer Science
Vrije Universiteit Brussel, VUB
Pleinlaan 2, B1050 Brussels, Belgium

dvermeir@vub.ac.be

ABSTRACT

Open answer set programming combines the strengths of logic programming (a rule-based presentation and a nonmonotonic semantics) and description logics (open domains). Reasoning under an open answer set semantics is undecidable in general, but decidability can be obtained for particular classes of logic programs, e.g., for bound guarded programs. In this paper, we show how bound guarded programs are expressive enough to simulate satisfiability checking in a DL with n-ary roles and nominals, yielding EXP-TIME-completeness for both the DL reasoning and the reasoning with bound guarded programs under the open answer set semantics. We establish decidability of three query problems (query containment, consistency, and disjointness) for *guarded* queries by a reduction to bound guarded programs, resulting in an EXPTIME upper bound.

Categories and Subject Descriptors

I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Logic Programming*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Theory

Keywords

Answer Set Programming, Open Domains, Description Logics

1. INTRODUCTION

Open answer set programming (OASP) [19, 17] combines the logic programming and first-order logic paradigms. From the former it inherits a rule-based presentation and a nonmonotonic semantics by means of negation as failure. In contrast with usual

logic programming semantics, see, e.g., the answer set semantics [12], OASP allows for domains consisting of other objects than those present in the logic program at hand. Such open domains are inspired by first-order logic based languages such as description logics (DLs) [2] and make OASP a viable candidate for conceptual reasoning. Due to their rule-based presentation and their support for nonmonotonic reasoning and open domains, OASP can be used to reason with both rule-based and conceptual knowledge on the Semantic Web, as illustrated in [19].

Take, for example, a logic program

$$
\begin{aligned}
p(X) &\leftarrow \quad not\ q(X) \\
q(a) &\leftarrow
\end{aligned}
$$

In normal answer set programming, one grounds the program with the program's constants, resulting in

$$
\begin{aligned}
p(a) &\leftarrow \quad not\ q(a) \\
q(a) &\leftarrow
\end{aligned}
$$

and the unique answer set $\{q(a)\}$. One concludes that p is not satisfiable as there is no answer set containing a p-atom. However, considering $p(X) \leftarrow not\ q(X)$ as a schema constraint and $q(a) \leftarrow$ as particular instance data, a knowledge engineer may want to check whether the first rule makes sense, i.e., can p be satisfied, independently of other instance data? To avoid that the engineer has to add all *significant* constants, the open answer set semantics allows for *open domains*. E.g., a *universe* $U = \{a, x\}$ where x is a new anonymous element. Grounding the example program with U gives the program

$$
\begin{aligned}
p(x) &\leftarrow \quad not\ q(x) \\
p(a) &\leftarrow \quad not\ q(a) \\
q(a) &\leftarrow
\end{aligned}
$$

which has an answer set $\{q(a), p(x)\}$. An *open answer set* of the program is $(U, \{q(a), p(x)\}$ where p is now satisfiable.

Formally, an open answer set in, e.g., [17] is a pair (U, M) where the *universe* U is a non-empty superset of the program's constants and M is the answer set of the program grounded with U. This definition takes on a *unique name assumption*, i.e., different constants are different element in the universe. As DL-based languages such as OWL DL [3] do not have such an assumption, we mend in this

paper the definition of universes. A universe becomes a pair (D, σ) where D is an arbitrary non-empty domain and σ maps the constants from the program to elements from D. This obliterates the unique name assumption from the original definition as $c_1 \neq c_2$ does not need to imply that $\sigma(c_1) \neq \sigma(c_2)$. Although we extend the definition of universe for compatibility with current Semantic Web ontology languages, note that the discussion on whether the unique name assumption is good or bad is far from settled, see, e.g., [8].

The main challenge for OASP is to control undecidability of satisfiability checking, a challenge it shares with DL-based languages. In [17], we identify a decidable class of programs, so-called *guarded programs*, for which decidability of satisfiability checking was obtained by a translation to guarded fixed point logic [14]. In this paper, we show the expressiveness of such guarded programs by simulating a DL with n-ary roles and nominals. In particular, we extend the DL $\mathcal{DLR}$ [6] with both *concept nominals* $\{o\}$ and *role nominals* $\{(o_1, \ldots, o_n)\}$, resulting in $\mathcal{DLRO}$. Due to the expressiveness of $\mathcal{DLR}$ one is able to simulate role nominals using concept nominals such that the former are just syntactic sugar. The DL $\mathcal{DLRO}$ with the number restrictions left out yields $\mathcal{DLRO}^{-\{\leq\}}$.

We translate satisfiability checking of $\mathcal{DLRO}^{-\{\leq\}}$ concept expressions w.r.t. $\mathcal{DLRO}^{-\{\leq\}}$ knowledge bases to satisfiability of predicates w.r.t. *bound* guarded programs. Since the latter is in EXPTIME [18] and the former is EXPTIME-hard, we deduce EXPTIME-completeness both for reasoning with $\mathcal{DLRO}^{-\{\leq\}}$ and with bound guarded programs. Previous simulations of DLs with logic programs under the open answer set semantics, e.g., [16, 19], considered only DLs with binary roles. Guarded programs allow for n-ary predicates and constants appearing arbitrarily throughout the program. As such they enable the simulation of DLs with n-ary roles and nominals, further dissolving the boundaries between DLs and logic programming. A current disadvantage of guarded programs is their inability of expressing number restrictions as is possible with $\mathcal{DLR}$. Translating $\mathcal{DLR}$ number restrictions yields programs that are not guarded, hence our restriction to $\mathcal{DLRO}^{-\{\leq\}}$.

We examine query problems such as *query containment*, *consistency*, and *disjointness* in the context of $\mathcal{DLRO}^{-\{\leq\}}$ schemas. We define *guarded* queries and show that the above query problems over $\mathcal{DLRO}^{-\{\leq\}}$ schemas can be reduced to satisfiability checking w.r.t. bound guarded programs. In [7], query containment w.r.t. $\mathcal{DLR}$ schemas is studied, demonstrating that query containment is undecidable in the general case where inequality is allowed in queries. The results presented in this paper show that one can allow for inequality in queries and still have decidability, provided the queries are guarded. In particular, query containment with guarded queries over $\mathcal{DLRO}^{-\{\leq\}}$ schemas is in EXPTIME.

The remainder of the paper starts with an introduction to open answer set programming in Section 2. Section 3 extends the DL $\mathcal{DLR}$ with nominals and leaves out number restrictions, resulting in $\mathcal{DLRO}^{-\{\leq\}}$, and Section 4 defines a reduction of $\mathcal{DLRO}^{-\{\leq\}}$ to bound guarded programs. In Section 5, we discuss three query problems and reduce them to satisfiability checking w.r.t. bound guarded programs. Finally, Section 6 points to related work. We conclude with directions for future research in Section 7.

2. DECIDABLE OPEN ANSWER SET PROGRAMMING

We introduce the open answer set semantics from [17], but modify it such that it does not take on a unique name assumption for constants by default. *Constants*, *variables*, *terms*, and *atoms* are

defined as usual. A *literal* is an atom $p(\vec{t})$ or a *naf-atom* $not\ p(\vec{t})$.[1] The *positive part* of a set of literals α is $\alpha^+ = \{p(\vec{t}) \mid p(\vec{t}) \in \alpha\}$ and the *negative part* of α is $\alpha^- = \{p(\vec{t}) \mid not\ p(\vec{t}) \in \alpha\}$. We assume the existence of binary predicates $=$ and $\neq$, where $t = s$ is considered as an atom and $t \neq s$ as $not\ t = s$. E.g., for $\alpha = \{X \neq Y, Y = Z\}$, we have $\alpha^+ = \{Y = Z\}$ and $\alpha^- = \{X = Y\}$. A *regular* atom is an atom that is not an equality atom. For a set X of atoms, $not\ X = \{not\ l \mid l \in X\}$.

A *program* is a countable set of rules $\alpha \leftarrow \beta$, where α and β are finite sets of literals, $|\alpha^+| \leq 1$, and $\forall t, s \cdot t = s \notin \alpha^+$, i.e., α contains at most one positive atom and this atom cannot be an equality atom.[2] The set α is the *head* of the rule and represents a disjunction of literals, while β is called the *body* and represents a conjunction of literals. If $\alpha = \emptyset$, the rule is called a *constraint*. *Free rules* are rules of the form $q(\vec{t}) \vee not\ q(\vec{t}) \leftarrow$ for a tuple $\vec{t}$ of terms; they enable a choice for the inclusion of atoms. We call a predicate p free if there is a free rule $p(\vec{t}) \vee not\ p(\vec{t}) \leftarrow$. Atoms, literals, rules, and programs that do not contain variables are *ground*.

For a literal, rule, or program o, let $cts(o)$ be the constants in o, $vars(o)$ its variables, and $preds(o)$ its predicates. A *universe* U for a program P is a pair (D, σ) where D is a non-empty *domain* and $\sigma : cts(P) \to D$ is a function that maps all constants to elements from D.[3] We call P_U the ground program obtained from P by substituting every variable in P by every possible element from D and every constant c by $\sigma(c)$. E.g., for a rule $r : p(X) \leftarrow f(X, c)$ and $U = (\{x, y\}, \sigma)$ where $\sigma(c) = x$, we have that the grounding w.r.t. U is

$$
\begin{aligned}
p(x) &\leftarrow f(x, x) \\
p(y) &\leftarrow f(y, x)
\end{aligned}
$$

Let $\mathcal{B}_P$ be the set of regular atoms that can be formed from a ground program P.

An *interpretation* I of a ground P is any subset of $\mathcal{B}_P$. For a ground regular atom $p(\vec{t})$, we write $I \models p(\vec{t})$ if $p(\vec{t}) \in I$; for an equality atom $p(\vec{t}) \equiv t = s$, we have $I \models p(\vec{t})$ if s and t are equal terms. We have $I \models not\ p(\vec{t})$ if $I \not\models p(\vec{t})$. For a set of ground literals X, $I \models X$ if $I \models l$ for every $l \in X$. A ground rule $r : \alpha \leftarrow \beta$ is *satisfied* w.r.t. I, denoted $I \models r$, if $I \models l$ for some $l \in \alpha$ whenever $I \models \beta$. Consequently, a ground constraint $\leftarrow \beta$ is satisfied w.r.t. I if $I \not\models \beta$. For a ground program P without *not*, an interpretation I of P is a *model* of P if I satisfies every rule in P; it is an *answer set* of P if it is a subset minimal model of P. For ground programs P containing *not*, the *GL-reduct* [12] w.r.t. I is defined as P^I, where P^I contains $\alpha^+ \leftarrow \beta^+$ for $\alpha \leftarrow \beta$ in P, $I \models not\ \beta^-$ and $I \models \alpha^-$. I is an *answer set* of a ground P if I is an answer set of P^I.

In the following, a program is assumed to be a finite set of rules; infinite programs only appear as byproducts of grounding a finite program with an infinite universe. An *open interpretation* of a program P is a pair (U, M) where U is a universe for P and M is an interpretation of P_U. An *open answer set* of P is an open interpretation (U, M) of P with M an answer set of P_U. An n-ary predicate p in P is *satisfiable* if there is an open answer set $((D, \sigma), M)$ of P and a $\vec{x} \in D^n$ such that $p(\vec{x}) \in M$.

---

[1] We have no classical negation $\neg$, however, programs with $\neg$ can be reduced to programs without it, see e.g. [24].

[2] The condition $|\alpha^+| \leq 1$ makes the GL-reduct non-disjunctive, ensuring that the *immediate consequence operator* is well-defined, see [17].

[3] A universe in [17] is a domain D that is a non-empty superset of the constants in P. This corresponds to a domain (D, σ) where σ is the identity function on D.

Satisfiability checking w.r.t. the open answer set semantics is undecidable in general. In [17], we identify a syntactically restricted fragment of programs, so-called *guarded programs*, for which satisfiability checking is decidable and obtained by a reduction to guarded fixed point logic [14]. Although the reduction in [17] assumes an open answer set semantics with a unique name assumption, it can be easily modified to fit the more general definition of universe we introduce in this section: leave out the guarded fixed point logic formulas that ensure that different constants are interpreted as different elements. The decidability of guarded programs relies on the presence of an atom in each rule that contains all variables of the rule, the *guard* of the rule. Formally, a rule $r : \alpha \leftarrow \beta$ is *guarded* if there is an atom $\gamma_b \in \beta^+$ such that $vars(r) \subseteq vars(\gamma_b)$; we call γ_b a *guard* of r. A program P is a *guarded program (GP)* if every non-free rule in P is guarded. E.g., a rule $a(X, Y) \leftarrow not\ f(X, Y)$ is not guarded, but $a(X, Y) \leftarrow g(X, Y), not\ f(X, Y)$ is guarded with guard $g(X, Y)$. Satisfiability checking of predicates w.r.t. guarded programs under the open answer set semantics is 2-EXPTIME-complete – a result that stems from the corresponding complexity in guarded fixed point logic.

Define the *width* of a fixed point logic formula ψ as the maximal number of free variables in its subformulas [13]. We define *bound* programs by looking at their corresponding first order form and the arity of its predicates. A program P is *bound* if every formula in $\texttt{sat}(P)$ is of bounded width and the predicates in P have a bounded arity, where $\texttt{sat}(P)$ is the first order form of P, i.e., $\bigwedge_{r:\alpha\leftarrow\beta\in P} \forall \vec{X} \cdot \beta \Rightarrow \alpha$, where $\vec{X}$ are the variables in r and *not* is replaced by $\neg$. In [18], we introduce those *bound guarded programs* and show that satisfiability checking is EXPTIME-complete. Note that the programs in [18] include *generalized literals* making them more expressive than the programs we consider in this paper. The EXPTIME-completeness for bound guarded programs with generalized literals yields an EXPTIME upper bound for bound guarded programs, but EXPTIME-hardness cannot be deduced directly (since the reduction in [18] uses generalized literals). In the next section however, we show EXPTIME-hardness for bound GPs without generalized literals as well.

We do not have a unique name assumption, i.e., it may be that $c_1 \neq c_2$ for constants c_1 and c_2 but that $\sigma(c_1) = \sigma(c_2)$ for a universe (D, σ). We can nonetheless impose the unique name assumption by adding $c_1 \neq c_2$ to the program for any pair of different constants. Such an addition is a quadratic transformation in the size of the original program and if the original program is a bound GP, the transformed one will be too.

3. EXTENDING $\mathcal{DLR}$ WITH CONCEPT AND ROLE NOMINALS

The DL $\mathcal{DLR}$ [6, 2] is a DL that supports n-ary roles, instead of the usual binary ones. We introduce $\mathcal{DLR}$ as in [2]. The basic building blocks in $\mathcal{DLR}$ are *concept names* A and *relation names* $\mathbf{P}$ where $\mathbf{P}$ denotes arbitrary n-ary relations for $2 \le n \le n_{max}$ and n_{max} is a given finite nonnegative integer. Role expressions $\mathbf{R}$ and concept expressions C can be formed according to the following syntax rules:

$$\mathbf{R} \rightarrow \top_n \mid \mathbf{P} \mid (\$i/n : C) \mid \neg\mathbf{R} \mid \mathbf{R_1} \sqcap \mathbf{R_2}$$
$$C \rightarrow \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid \le k[\$i]\mathbf{R}$$

where we assume i is between 1 and n in $(\$i/n : C)$, and similarly in $\exists[\$i]\mathbf{R}$ and $\le k[\$i]\mathbf{R}$ if $\mathbf{R}$ is an n-ary relation. Moreover, we assume that the above constructs are well-typed, e.g., $\mathbf{R_1} \sqcap \mathbf{R_2}$ is defined only for relations of the same arity. The semantics of $\mathcal{DLR}$ is given by interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ is a non-

empty set, the *domain*, and $\cdot^{\mathcal{I}}$ is an interpretation function such that $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, $\mathbf{R}^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$ for an n-ary relation $\mathbf{R}$, and the following conditions are satisfied ($\mathbf{P}, \mathbf{R}, \mathbf{R_1}$, and $\mathbf{R_2}$ have arity n):

$$\top_n^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$$
$$\mathbf{P}^{\mathcal{I}} \subseteq \top_n^{\mathcal{I}}$$
$$(\neg\mathbf{R})^{\mathcal{I}} = \top_n^{\mathcal{I}} \backslash \mathbf{R}^{\mathcal{I}}$$
$$(\mathbf{R_1} \sqcap \mathbf{R_2})^{\mathcal{I}} = \mathbf{R_1}^{\mathcal{I}} \cap \mathbf{R_2}^{\mathcal{I}}$$
$$(\$i/n : C)^{\mathcal{I}} = \{(d_1, \dots, d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\}$$
$$\top_1^{\mathcal{I}} = \Delta^{\mathcal{I}}$$
$$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$$
$$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$$
$$(\exists[\$i]\mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists(d_1, \dots, d_n) \in \mathbf{R}^{\mathcal{I}} \cdot d_i = d\}$$
$$(\le k[\$i]\mathbf{R})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid |\{(d_1, \dots, d_n) \in \mathbf{R}^{\mathcal{I}} \mid d_i = d\}| \le k\}$$

Note that in $\mathcal{DLR}$ the negation of role expressions is defined w.r.t. $\top_n^{\mathcal{I}}$ instead of w.r.t. $(\Delta^{\mathcal{I}})^n$. A $\mathcal{DLR}$ knowledge base consists of terminological axioms and role axioms defining subset relations between concept expressions and role expressions (of the same arity) respectively. A terminological axiom $C_1 \sqsubseteq C_2$ is *satisfied* by $\mathcal{I}$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. A role axiom $\mathbf{R_1} \sqsubseteq \mathbf{R_2}$ is *satisfied* by $\mathcal{I}$ iff $\mathbf{R_1}^{\mathcal{I}} \subseteq \mathbf{R_2}^{\mathcal{I}}$. An interpretation is a *model* of a knowledge base if all axioms are satisfied by the interpretation. A concept expression C is *satisfiable* w.r.t. a knowledge base Σ if there is a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$.

We extend $\mathcal{DLR}$ with nominals $\{o\}$ and $\{(o_1, \dots, o_n)\}$ and call the resulting DL $\mathcal{DLRO}$:

$$\mathbf{R} \rightarrow \top_n \mid \mathbf{P} \mid (\$i/n : C) \mid \neg\mathbf{R} \mid \mathbf{R_1} \sqcap \mathbf{R_2} \mid \{(o_1, \dots, o_n)\}$$
$$C \rightarrow \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid \le k[\$i]\mathbf{R} \mid \{o\}$$

We distinguish between *concept* nominals $\{o\}$ and (n-ary) *role* nominals $\{(o_1, \dots, o_n)\}$ for $n > 1$. The former are concept expressions while the latter are role expressions. We interpret concept nominals as singleton subsets of $\Delta^{\mathcal{I}}$ and role nominals as singleton subsets of $\top_n^{\mathcal{I}}$. The definition of an interpretation $\mathcal{I}$ is extended such that for any b in some nominal $b^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, i.e., $\mathcal{I}$ maps b to an element in the domain. The semantics of nominals satisfies then the following constraints:

$$\{o\}^{\mathcal{I}} = \{o^{\mathcal{I}}\} \subseteq \Delta^{\mathcal{I}}$$
$$\{(o_1, \dots, o_n)\}^{\mathcal{I}} = \{(o_1^{\mathcal{I}}, \dots, o_n^{\mathcal{I}})\} \subseteq \top_n^{\mathcal{I}}$$

Although nominals are interpreted as singleton sets, one can use them to express sets of *named individuals/tuples* by using other constructs, e.g., $\{o_1, o_2\} \equiv \{o_1\} \sqcup \{o_2\}$.

Define a *Cartesian product* role expression $C_1 \times \dots \times C_n$ for concept expressions $C_1, \dots, C_n$ with the following semantics:

$$(C_1 \times \dots \times C_n)^{\mathcal{I}} \equiv C_1^{\mathcal{I}} \times \dots \times C_n^{\mathcal{I}} ,$$

where $\mathcal{I}$ is an interpretation and the $\times$ on the right hand side is the usual Cartesian products of sets. Cartesian product role expressions do not add extra expressiveness to $\mathcal{DLR}$ as they can be rewritten in terms of $\mathcal{DLR}$ constructs only. For every interpretation $\mathcal{I}$, one has

$$(C_1 \times \dots \times C_n)^{\mathcal{I}} = ((\$1/n : C_1) \sqcap \dots \sqcap (\$n/n : C_n))^{\mathcal{I}} .$$

Since the interpretation of role nominals can be seen as a Cartesian product of the interpretation of concept nominals, role nominals do

not add any expressive power either: $\{(o_1, \ldots, o_n)\}$ is equivalent with $(\$1/n : \{o_1\}) \sqcap \ldots \sqcap (\$n/n : \{o_n\})$.

THEOREM 1. *Let $\mathcal{I}$ be an interpretation. Then,*

$$\{(o_1, \ldots, o_n)\}^{\mathcal{I}} = ((\$1/n : \{o_1\}) \sqcap \ldots \sqcap (\$n/n : \{o_n\}))^{\mathcal{I}} .$$

PROOF. Immediate by the definition of role nominals and the above remark on cardinality product role expressions. □

As the rewriting in Theorem 1 is linear, we can restrict ourselves in the remainder of the paper to concept nominals only.

We denote the fragment of $\mathcal{DLRO}$ without the number restriction $\leq k[\$i]\mathbf{R}$ as $\mathcal{DLRO}^{-\{\leq\}}$. As $\mathcal{DLRO}^{-\{\leq\}}$ is an extension of the DL $\mathcal{ALC}$ and satisfiability checking of $\mathcal{ALC}$ concept expressions w.r.t. $\mathcal{ALC}$ knowledge bases is EXPTIME-complete [2], we have that satisfiability checking of $\mathcal{DLRO}^{-\{\leq\}}$ concept expressions w.r.t. $\mathcal{DLRO}^{-\{\leq\}}$ knowledge bases is EXPTIME-hard.

4. SIMULATING $\mathcal{DLRO}^{-\{\leq\}}$ WITH BOUND GUARDED PROGRAMS

Consider a knowledge base Σ where L and M are concepts and $\mathbf{t}$ is a ternary role such that, intuitively, (x, y, z) is in the interpretation of $\mathbf{t}$ if x is a parent of y which in turn is a parent of z. We interpret L as the concept indicating the *local* people of the village M.

$$L \sqsubseteq M \sqcap \exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M)) .$$

The axiom expresses that local people live in M and have a parent and grandparent that live in M.

We translate the axiom as a constraint[4]:

$$\leftarrow L(X),$$
$$not\ (M \sqcap \exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M)))(X)$$

Intuitively, we associate with the concept expressions on either side of $\sqsubseteq$ in an axiom a new predicate name. We conveniently denote this new predicate like the corresponding concept expression. The constraints simulate the behavior of the axioms: if $L(x)$ holds and $(M \sqcap \exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M)))(x)$ does not, we have a contradiction. This corresponds to the DL behavior of the corresponding axiom: if $x \in L^{\mathcal{I}}$ and $x \notin (M \sqcap \exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M)))^{\mathcal{I}}$, we have a contradiction as the axiom requires that $L^{\mathcal{I}} \subseteq (M \sqcap \exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M)))^{\mathcal{I}}$ for models $\mathcal{I}$.

After translating the axiom as a constraint, it remains to define the newly introduced predicates according to the DL semantics. We define L, M, and $\mathbf{t}$ as free predicates:

$$L(X) \vee not\ L(X) \leftarrow$$
$$M(X) \vee not\ M(X) \leftarrow$$
$$\mathbf{t}(X, Y, Z) \vee not\ \mathbf{t}(X, Y, Z) \leftarrow$$

Intuitively, the DL semantics gives an *open* (first-order) interpretation to its concept names and role names: a (tuple of) domain element(s) is either in the interpretation of a concept (role) name or not. Note that $\mathbf{t}$ is a ternary role name, so we introduce it as a ternary predicate.

We add for the special concept name $\top_1$ and $\top_3$ the following free rules

$$\top_1(X) \vee not\ \top_1(X) \leftarrow$$
$$\top_3(X, Y, Z) \vee not\ \top_3(X, Y, Z) \leftarrow$$

[4]We assume that a logic program may contain predicate names starting with a capital letter; this should not lead to confusion with variables, which appear only as arguments of predicates.

We did not introduce a rule for $\top_2$ since the example knowledge base does not contain binary roles. We ensure that $\mathbf{t}^{\mathcal{I}} \subseteq \top_3^{\mathcal{I}}$ by a constraint

$$\leftarrow \mathbf{t}(X, Y, Z), not\ \top_3(X, Y, Z)$$

which is guarded by $\mathbf{t}(X, Y, Z)$. To ensure that $\top_1^{\mathcal{I}} = \Delta^{\mathcal{I}}$, we add the constraint

$$\leftarrow not\ \top_1(X)$$

For rules containing only one variable we can always assume that $X = X$ is in the body and acts as the guard of the rule.

The predicate $(M \sqcap \exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M)))$ can be defined by the rule

$$(M \sqcap \exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M)))(X) \leftarrow M(X),$$
$$\exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))(X)$$

Intuitively, if $(M \sqcap \exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M)))(x)$ is in an open answer set, then, by minimality of open answer sets, $M(x)$ and $\exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))(x)$ have to be in the open answer set. Vice versa, if $M(x)$ and $\exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))(x)$ are in the open answer set, then $(M \sqcap \exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M)))(x)$ has to be as well since the rules must be satisfied. This corresponds exactly to the DL semantics for concept conjunction.

We define the predicate $\exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))$ by the rule

$$\exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))(Z) \leftarrow$$
$$(\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))(X, Y, Z)$$

such that $\exists[\$3](\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))(z)$ holds in an open answer set iff there are some x and y such that $(\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))(x, y, z)$ holds. The ternary predicate $(\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))$ represents a conjunction and is defined by the rule

$$(\mathbf{t} \sqcap (\$1/3 : M) \sqcap (\$2/3 : M))(X, Y, Z) \leftarrow \mathbf{t}(X, Y, Z),$$
$$(\$1/3 : M)(X, Y, Z), (\$2/3 : M)(X, Y, Z)$$

Finally, we define the predicates $(\$1/3 : M)$ and $(\$2/3 : M)$ as follows:

$$(\$1/3 : M)(X, Y, Z) \leftarrow \top_3(X, Y, Z), M(X)$$
$$(\$2/3 : M)(X, Y, Z) \leftarrow \top_3(X, Y, Z), M(Y)$$

Both rules are guarded by $\top_3(X, Y, Z)$. $(\$1/3 : M)(x, y, z)$ holds in an open answer set iff $\top_3(x, y, z)$ holds and $M(x)$ holds which corresponds to the definition of $(x, y, z) \in (\$1/3 : M)^{\mathcal{I}}$ for an interpretation $\mathcal{I}$.

We define the formal translation from $\mathcal{DLRO}^{-\{\leq\}}$ satisfiability checking to satisfiability checking w.r.t. bound GPs using the notion of *closure*. Define the *closure* $clos(C, \Sigma)$ of a $\mathcal{DLRO}^{-\{\leq\}}$ concept expression C and a $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base Σ as the smallest set satisfying the following conditions:

- $C \in clos(C, \Sigma)$,

- $\top_1 \in clos(C, \Sigma)$,

- for each $C \sqsubseteq D$ an axiom in Σ (role or terminological), $\{C, D\} \subseteq clos(C, \Sigma)$,

- for every D in $clos(C, \Sigma)$, $clos(C, \Sigma)$ should contain every subformula that is a concept expression or a role expression,

- if $clos(C, \Sigma)$ contains n-ary relation names, it must contain $\top_n$.

Formally, we define $\Phi(C, \Sigma)$ to be the following bound GP, obtained from the $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base Σ and the concept expression C:

- For each terminological axiom $C \sqsubseteq D \in \Sigma$, add the constraint

$$\leftarrow C(X), not\ D(X) \qquad (1)$$

- For each role axiom $\mathbf{R} \sqsubseteq \mathbf{S} \in \Sigma$ where $\mathbf{R}$ and $\mathbf{S}$ are n-ary, add the constraint

$$\leftarrow \mathbf{R}(X_1, \ldots, X_n), not\ \mathbf{S}(X_1, \ldots, X_n) \qquad (2)$$

- For each $\top_n \in clos(C, \Sigma)$, add the free rule

$$\top_n(X_1, \ldots, X_n) \vee not\ \top_n(X_1, \ldots, X_n) \leftarrow \qquad (3)$$

Furthermore, for each n-ary relation name $\mathbf{P} \in clos(C, \Sigma)$, we add the constraint

$$\leftarrow \mathbf{P}(X_1, \ldots, X_n), not\ \top_n(X_1, \ldots, X_n) \qquad (4)$$

Intuitively, the latter rule ensures that $\mathbf{P}^{\mathcal{I}} \subseteq \top_n^{\mathcal{I}}$. We add a constraint

$$\leftarrow not\ \top_1(X) \qquad (5)$$

which enforces that for every element x in the universe, $\top_1(x)$ is true in the open answer set. The latter rule ensures that $\top_1^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for the corresponding interpretation. The rule can be guarded with $X = X$.

- Next, we distinguish between the types of concept and role expressions that appear in $clos(C, \Sigma)$. For $D \in clos(C, \Sigma)$:

  - if D is a concept nominal $\{o\}$, add

$$D(o) \leftarrow \qquad (6)$$

  This will ensure that $\{o\}(x)$ holds in an open answer set iff $x = \sigma(o)$, corresponding to $\{o\}^{\mathcal{I}} = \{x\}$ for an interpretation $\mathcal{I}$.

  - if D is a concept name, add

$$D(X) \vee not\ D(X) \leftarrow \qquad (7)$$

  - if $\mathbf{D}$ is an n-ary relation name, add

$$\mathbf{D}(X_1, \ldots, X_n) \vee not\ \mathbf{D}(X_1, \ldots, X_n) \leftarrow \qquad (8)$$

  - if $D = \neg E$ for a concept expression E, add

$$D(X) \leftarrow not\ E(X) \qquad (9)$$

  Note that we can assume that such a rule is guarded by $X = X$.

  - if $D = \neg\mathbf{R}$ for an n-ary role expression $\mathbf{R}$, add

$$D(X_1, \ldots, X_n) \leftarrow \\ \top_n(X_1, \ldots, X_n), not\ \mathbf{R}(X_1, \ldots, X_n) \qquad (10)$$

  Note that if negation was defined w.r.t. to $(\Delta^{\mathcal{I}})^n$ instead of $\top_n^{\mathcal{I}}$, we would not be able to write the above as a guarded rule.

  - if $D = E \sqcap F$ for concept expressions E and F, add

$$D(X) \leftarrow E(X), F(X) \qquad (11)$$

  - if $D = \mathbf{E} \sqcap \mathbf{F}$ for n-ary role expressions $\mathbf{E}$ and $\mathbf{F}$, add

$$D(X_1, \ldots, X_n) \leftarrow \mathbf{E}(X_1, \ldots, X_n), \mathbf{F}(X_1, \ldots, X_n) \qquad (12)$$

  - if $D = (\$i/n : C)$, add

$$D(X_1, \ldots, X_i, \ldots, X_n) \leftarrow \\ \top_n(X_1, \ldots, X_i, \ldots, X_n), C(X_i) \qquad (13)$$

  - if $D = \exists[\$i]\mathbf{R}$, add

$$D(X) \leftarrow \mathbf{R}(X_1, \ldots, X_{i-1}, X, X_{i+1}, \ldots, X_n) \qquad (14)$$

THEOREM 2. *Let Σ be a $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base and C a $\mathcal{DLRO}^{-\{\leq\}}$ concept expression. Then, $\Phi(C, \Sigma)$ is a bound GP, with a size that is polynomial in the size of C and Σ.*

PROOF. Observing the rules in $\Phi(C, \Sigma)$, it is clear that this program is a GP. Furthermore, every rule contains at most n_{max} variables which is also the bound for the arity of the predicates such that $\Phi(C, \Sigma)$ is a bound GP.

The size of $clos(C, \Sigma)$ is linear in C and Σ. The size of the GP $\Phi(C, \Sigma)$ is polynomial in the size of $clos(C, \Sigma)^5$ such that the result follows. $\square$

THEOREM 3. *A $\mathcal{DLRO}^{-\{\leq\}}$ concept expression C is satisfiable w.r.t. a $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base Σ iff the predicate C is satisfiable w.r.t. $\Phi(C, \Sigma)$.*

PROOF. For the "only if" direction, assume the concept expression C is satisfiable w.r.t. Σ, i.e., there exists a model $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ with $C^{\mathcal{I}} \neq \emptyset$. We construct the open answer set $((U, \sigma), M)$ with $U \equiv \Delta^{\mathcal{I}}$, $\sigma(o) \equiv o^{\mathcal{I}}$, and

$$M \equiv \{C(x) \mid x \in C^{\mathcal{I}}, C \in clos(C, \Sigma)\}$$
$$\cup \{\mathbf{R}(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in \mathbf{R}^{\mathcal{I}}, R \in clos(C, \Sigma)\}$$

with C and $\mathbf{R}$ concept expressions and role expressions respectively.

We have that $((U, \sigma), M)$ is an open answer set of $\Phi(C, \Sigma)$ that satisfies C:

1. Since $C^{\mathcal{I}} \neq \emptyset$ there clearly is an $x \in U$ such that $C(x) \in M$.

2. One can check that M is a minimal model of $\Phi(C, \Sigma)^M_{(U, \sigma)}$.

For the "if" direction, assume $((U, \sigma), M)$ is an open answer set of $\Phi(C, \Sigma)$ with $C(u) \in M$. Define an interpretation $\mathcal{I} \equiv (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, with $\Delta^{\mathcal{I}} \equiv U$, $A^{\mathcal{I}} \equiv \{x \mid A(x) \in M\}$ for concept names A, $\mathbf{R}^{\mathcal{I}} \equiv \{(x_1, \ldots, x_n) \mid \mathbf{R}(x_1, \ldots, x_n) \in M\}$ for n-ary role names $\mathbf{R}$ and $o^{\mathcal{I}} = \sigma(o)$, for $o \in cts(\Phi(C, \Sigma))$.

One can show that $x \in D^{\mathcal{I}}$ iff $D(x) \in M$ for a concept expression D and $(x_1, \ldots, x_n) \in \mathbf{R}^{\mathcal{I}}$ iff $\mathbf{R}(x_1, \ldots, x_n) \in M$, for an n-ary role expression $\mathbf{R}$.

---

[5]The size of $\Phi(C, \Sigma)$ is polynomial in the size of $clos(C, \Sigma)$ provided the size of C and Σ increases such that the n in an added n-ary role expression is polynomial in the size of the maximal arity of role expressions in C and Σ. Although the size of C and Σ increases linearly upon adding a relation name $\mathbf{R}$ with arity 2^n, where n is the maximal arity of relation names in C and Σ, the size of $\Phi(C, \Sigma)$ increases exponentially: one needs to add, e.g., rules

$$\top_{2^n}(X_1, \ldots, X_{2^n}) \vee not\ \top_{2^n}(X_1, \ldots, X_{2^n}) \leftarrow$$

One can check that $\mathcal{I}$ satisfies every terminological axiom $D_1 \sqsubseteq D_2$ as well as every role axiom.

It remains to check that $C^{\mathcal{I}}$ is not empty. We have that $C(u) \in M$ and we know that this is only possible, by the above, if $u \in C^{\mathcal{I}}$. $\square$

In the above, we considered the fragment $\mathcal{DLRO}^{-\{\leq\}}$ of $\mathcal{DLRO}$ without the expressions $\leq k[\$i]\mathbf{R}$ since such expressions cannot be simulated with guarded programs. E.g., consider the concept expression $\leq 1[\$1]R$ where R is a binary role. One can simulate the $\leq$ by negation as failure:

$$\leq 1[\$1]R(X) \leftarrow not\ q(X)$$

for some new q with q defined such that there are at least 2 different R-successors:

$$q(X) \leftarrow R(X, Y_1), R(X, Y_2), Y_1 \neq Y_2$$

However, the latter rule is not a guarded rule – there is no atom that contains X, Y_1, and Y_2. So, in general, expressing number restrictions such as $\leq k[\$i]\mathbf{R}$ is out of reach for guarded programs.

Since satisfiability checking w.r.t. bound GPs is in EXPTIME, Theorems 2 and 3 ensure that satisfiability checking of $\mathcal{DLRO}^{-\{\leq\}}$ concept expressions w.r.t. $\mathcal{DLRO}^{-\{\leq\}}$ knowledge bases is in EXPTIME. Together with the EXPTIME-hardness, $\mathcal{DLRO}^{-\{\leq\}}$ satisfiability checking w.r.t. $\mathcal{DLRO}^{-\{\leq\}}$ knowledge bases is EXPTIME-complete.

THEOREM 4. *Satisfiability checking of $\mathcal{DLRO}^{-\{\leq\}}$ concepts w.r.t. $\mathcal{DLRO}^{-\{\leq\}}$ knowledge bases is* EXPTIME-*complete.*

From [18], we know that satisfiability checking of predicates w.r.t. bound GPs is in EXPTIME. From Theorems 3 and 4, we have that reasoning is EXPTIME-hard.

THEOREM 5. *Satisfiability checking of predicates w.r.t. bound GPs is* EXPTIME-*complete.*

5. QUERY PROBLEMS OVER $\mathcal{DLRO}^{-\{\leq\}}$ SCHEMAS

Consider a $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base Σ consisting of the axiom[6]

$$R \equiv (\$1/2 : A) .$$

We can query Σ for those x's for which A holds and that have an R successor, i.e., we define a *query*

$$q_1(X) \equiv A(X), R(X, Y) .$$

We call q_1 a query over the *schema* Σ. Formally, we define, similar to [7], a *schema* as a $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base and a *query q* over a schema Σ as an expression of the form

$$q(\vec{X}) \equiv body_1(\vec{X}, \vec{Y}_1, \vec{c}_1) \vee \ldots \vee body_m(\vec{X}, \vec{Y}_m, \vec{c}_m) , \quad (15)$$

where $\vec{X}, \vec{Y}_i$, and $\vec{c}_i$ subsume the variables and constants in the expression $body_i(\vec{X}, \vec{Y}_i, \vec{c}_i)$. The latter expression is a conjunction of literals (i.e., atoms and classically negated atoms) where each underlying predicate is a concept or role name from Σ.[7] We call the query n-ary if $\vec{X}$ is an n-ary sequence of variables.

---

[6] $A \equiv B$ is shorthand for the axioms $A \sqsubseteq B$ and $B \sqsubseteq A$.

[7] We do not allow for arbitrary concept expressions or role expressions C in a query. However, to obtain the same effect one can easily add $A \equiv C$, for a concept (role) name A, to the knowledge base and use an atom with underlying predicate A in the query.

The semantics of an n-ary query q over a schema Σ is given using an interpretation $\mathcal{I}$ of Σ:

$$q^{\mathcal{I}} \equiv \{\vec{o} \in (\Delta^{\mathcal{I}})^n \mid \exists i \cdot \exists \vec{y}_i \in (\Delta^{\mathcal{I}})^{k_i} \cdot body_i(\vec{o}, \vec{y}_i, \vec{c}_i) \text{ true in } \mathcal{I}\} ,$$

where k_i is the length of $\vec{Y}_i$. A ground atom $A(\vec{t}\,)$ is *true in $\mathcal{I}$* if $\vec{t}^{\mathcal{I}} \in A^{\mathcal{I}}$ where $\vec{t}^{\mathcal{I}}$ is $\vec{t}$ with all constants c replaced by $c^{\mathcal{I}}$. If c is present in Σ, $c^{\mathcal{I}}$ is well-defined, otherwise we assume $\mathcal{I}$ is extended and has defined $c^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. A negated atom $\neg a$ is true in $\mathcal{I}$ if a is not true in $\mathcal{I}$, and a conjunction is true if all conjuncts are true.

We focus on three query problems for schemas: query containment, consistency, and disjointness, identified in [2] as important reasoning tasks in the context of databases that have to satisfy a conceptual schema.

Query Containment. The query containment problem involves checking, given two queries q_1 and q_2, whether, for every database satisfying a schema Σ, the extension of q_1 is a subset of the extension of q_2. Formally, we say, for a fixed schema Σ, that q_1 *is contained in q_2*, denoted $q_1 \sqsubseteq q_2$, iff for all models $\mathcal{I}$ of Σ, $q_1^{\mathcal{I}} \subseteq q_2^{\mathcal{I}}$. We denote $q_1 \equiv q_2$ if both $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$.

In addition to the above example query q_1, define the query

$$q_2(X) \equiv R(X, Y) .$$

By definition of R in Σ, we have that $q_1 \equiv q_2$. This illustrates the usefulness of query containment. The knowledge engineer can replace all queries q_1 by the query q_2 which are likely to be faster since they have smaller bodies.

Definition 1. Let q be an n-ary query as in (15) over a schema Σ. Then, q is *guarded* if

- every $body_i(\vec{X}, \vec{Y}_i, \vec{c}_i)$ contains an atom a_i s.t. $vars(a_i) = vars(body_i(\vec{X}, \vec{Y}_i, \vec{c}_i))$,

- $n \leq n_{max}$, and

- for k_i the length of $\vec{Y}_i$, $k_i \leq n_{max}$.

We can reduce query containment for guarded queries over a schema to satisfiability checking w.r.t. bound GPs. For an n-ary query q as in (15) define $\psi(q)$ as the following rules:

$$\psi(q) \equiv \begin{cases} q(\vec{X}) \leftarrow body_1(\vec{X}, \vec{Y}_1, \vec{c}_1) \\ \vdots \\ q(\vec{X}) \leftarrow body_m(\vec{X}, \vec{Y}_m, \vec{c}_m) \end{cases}$$

where $body_i(\vec{X}, \vec{Y}_i, \vec{c}_i)$ is the flattened list of atoms and naf-atoms making up its conjunction, e.g., if $body_i(\vec{X}, \vec{Y}_i, \vec{c}_i) = a(X) \wedge \neg b(X)$, then we write this as $a(X), not\ b(X)$. Intuitively, the disjunction in the query is taken care of by the different rules with the same head $q(\vec{X})$ while conjunction is standard conjunction in the body of the rule and $\neg$ is replaced by *not*. For a schema Σ and a set of queries Q over Σ, define the program $\psi(Q, \Sigma)$ as follows:

$$\psi(Q, \Sigma) \equiv \{\psi(q) \mid q \in Q\} \cup \Phi(\top_1, \Sigma) .$$

The program $\psi(Q, \Sigma)$ is the program obtained from translating the schema Σ as in Theorem 3 and the queries Q according the above correspondence ψ for queries.

THEOREM 6. *Let Σ be a schema, q_1 and q_2 guarded queries over Σ, and q a new predicate. Then $q_1 \sqsubseteq q_2$ iff q is unsatisfiable w.r.t. $\psi(\{q_1, q_2\}, \Sigma) \cup \{q(\vec{X}) \leftarrow q_1(\vec{X}), not\ q_2(\vec{X})\}$, where the latter is a bound GP.*

PROOF. For the "only if" direction, take $q_1 \sqsubseteq q_2$, where

$$q_1(\vec{X}) \equiv body_1^1(\vec{X}, \vec{Y}_1^1, \vec{c}_1^1) \vee \ldots \vee body_m^1(\vec{X}, \vec{Y}_m^1, \vec{c}_m^1)$$

and

$$q_2(\vec{X}) \equiv body_1^2(\vec{X}, \vec{Y}_1^2, \vec{c}_1^2) \vee \ldots \vee body_m^2(\vec{X}, \vec{Y}_m^2, \vec{c}_m^2).$$

By contradiction, take q satisfiable w.r.t. $P \equiv \psi(\{q_1, q_2\}, \Sigma) \cup \{q(\vec{X}) \leftarrow q_1(\vec{X}), not\ q_2(\vec{X})\}$. Then there is an open answer set $((U, \sigma), M)$ of P such that $q(\vec{x}) \in M$, and consequently, by minimality of open answer sets, $q_1(\vec{x}) \in M$ and $q_2(\vec{x}) \notin M$. Again by minimality of open answer sets, we have that there is an i and a $\vec{y}_i^1 \in U^{k_i^1}$ such that $M \models body_i^1(\vec{x}, \vec{y}_i^1, \sigma(\vec{c}_i^1))$. Furthermore, for all i and for all $\vec{y}_i^2 \in U^{k_i^2}$, $M \not\models body_i^2(\vec{x}, \vec{y}_i^2, \sigma(\vec{c}_i^2))$. Define $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $\Delta^{\mathcal{I}} \equiv U$, $A^{\mathcal{I}} \equiv \{x \mid A(x) \in M\}$ for concept names A, $\mathbf{R}^{\mathcal{I}} \equiv \{(x_1, \ldots, x_n) \mid \mathbf{R}(x_1, \ldots, x_n) \in M\}$ for n-ary role names $\mathbf{R}$ and $o^{\mathcal{I}} = \sigma(o)$, for $o \in cts(P)$. One can show that $\mathcal{I}$ is a model of Σ such that $\vec{x} \in q_1^{\mathcal{I}} \setminus q_2^{\mathcal{I}}$, a contradiction with $q_1 \sqsubseteq q_2$.

For the "if" direction, take q unsatisfiable w.r.t. P and $\mathcal{I}$ a model of Σ that defines the constants in q_1 and q_2. Assume, by contradiction, there is an $\vec{x}$ such that $\vec{x} \in q_1^{\mathcal{I}} \setminus q_2^{\mathcal{I}}$. Define $((U, \sigma), M)$ such that $U \equiv \Delta^{\mathcal{I}}$, $\sigma(c) = c^{\mathcal{I}}$ for constants in P and

$$M \equiv \{C(x) \mid x \in C^{\mathcal{I}}, C \in clos(\top_1, \Sigma)\}$$

$$\cup \{\mathbf{R}(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in \mathbf{R}^{\mathcal{I}}, \mathbf{R} \in clos(\top_1, \Sigma)\}$$

$$\cup \{q(\vec{x}) \mid \vec{x} \in q_1^{\mathcal{I}} \setminus q_2^{\mathcal{I}}\} \cup \{q_1(\vec{x}) \mid \vec{x} \in q_1^{\mathcal{I}}\} \cup \{q_2(\vec{x}) \mid \vec{x} \in q_2^{\mathcal{I}}\}.$$

One can show that $((U, \sigma), M)$ is an open answer set of P with $q(\vec{x}) \in M$, a contradiction with the unsatisfiability of q w.r.t. P. $\Phi(\top_1, \Sigma)$ is a bound GP by Theorem 2. Moreover, $\psi(q_1)$ and $\psi(q_2)$ are bound GPs as well, since q_1 and q_2 are guarded. Finally, $q(\vec{X}) \leftarrow q_1(\vec{X}), not\ q_2(\vec{X})$ is guarded by $q_1(\vec{X})$; it is bound since q has the same arity as q_1 and q_2 and the latter are guarded queries. $\square$

Query Consistency. This involves checking whether, for a given query, there is a database satisfying the schema, such that the extension of the query predicate is not empty. Formally, a query q over a schema Σ is *consistent* if there exists a model $\mathcal{I}$ of Σ such that $q^{\mathcal{I}} \neq \emptyset$.

THEOREM 7. *Let Σ be a schema and q a guarded query over Σ. Then q is consistent iff q is satisfiable w.r.t. $\psi(\{q\}, \Sigma)$, where the latter is a bound GP.*

PROOF. Similarly as Theorem 6. $\square$

Query Disjointness. Are two queries q_1 and q_2 disjoint for every database satisfying the schema? Formally, q_1 *is disjoint from* q_2 over a schema Σ iff for all models $\mathcal{I}$ of Σ, $q_1^{\mathcal{I}} \cap q_2^{\mathcal{I}} = \emptyset$. This can again be checked by a reduction to satisfiability checking w.r.t. bound GPs.

THEOREM 8. *Let Σ be a schema, q_1 and q_2 guarded queries over Σ, and q a new predicate. Then q_1 is disjoint from q_2 iff q is unsatisfiable w.r.t. $\psi(\{q_1, q_2\}, \Sigma) \cup \{q(\vec{X}) \leftarrow q_1(\vec{X}), q_2(\vec{X})\}$, where the latter is a bound GP.*

PROOF. Similarly as Theorem 6. $\square$

As a consequence of the reductions of those query problems to satisfiability checking w.r.t. bound GPs we have, by Theorem 4, that query containment and disjointness are in CO-EXPTIME and query consistency is in EXPTIME.

THEOREM 9. *Query containment and disjointness for guarded queries over $\mathcal{DLRO}^{-\{\leq\}}$ schemas are in CO-EXPTIME. Query consistency for guarded queries over $\mathcal{DLRO}^{-\{\leq\}}$ schemas is in EXPTIME.*

6. RELATED WORK

We distinguish between two lines of research involving the reconciliation of DLs and logic programming paradigms: the approach that tries to simulate DLs reasoning with logic programming by taking a DL knowledge base and reducing it to a program such that both conclude the same regarding satisfiability checking (see, e.g., [15, 1, 22, 28, 4]) and the approach that unites the strengths of DLs and LP by letting them coexist and interact, but without reducing one formalism to the other per se (see, e.g., [23, 25, 9, 26, 27, 10, 11, 21]). For more details on those lines of research and their relation to open answer set programming, we refer to [20, 16, 19].

Note that all the above approaches simulate DLs that only allow for concept expressions and binary roles; we simulated $\mathcal{DLRO}^{-\{\leq\}}$ which allows for n-ary roles. In Theorem 3 of [7] the undecidable unbounded tiling problem [5] is reduced to query containment of queries with inequality over a schema that falls strictly[8] in the $\mathcal{DLRO}^{-\{\leq\}}$ language. This leads to the observation that query containment with queries containing inequality over $\mathcal{DLRO}^{-\{\leq\}}$ schemas is undecidable in general. Theorem 6 shows that queries may contain inequality without inducing undecidability of query containment provided the queries are guarded. *Guardedness* is thus a sufficient condition for decidability of query containment, regardless of the presence of inequality atoms. This sufficiency is consistent with the non-guardedness of the constructed queries in Theorem 3 of [7].

7. CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

We extended $\mathcal{DLR}$ with nominals and reduced satisfiability checking of $\mathcal{DLRO}^{-\{\leq\}}$ concept expressions w.r.t. $\mathcal{DLRO}^{-\{\leq\}}$ knowledge bases to satisfiability checking w.r.t. bound guarded programs. This yielded EXPTIME-completeness for both reasoning problems. We defined guarded queries and demonstrated that the guardedness is sufficient for decidability of query containment, consistency, and disjointness, again by a reduction to bound guarded programs. DLs are a current standard for reasoning on the web and there is an increasing interest in extending DLs with rule-based reasoning. The results in this paper present an integrated approach to such DLs and rule-based reasoning.

A significant disadvantage of the current bound guarded programs is its lack of support for number restrictions in $\mathcal{DLRO}$. One approach to mend this is looking at possible extensions to guarded fixed point logic, in particular, can one allow a looser form of inequality in guarded fixed point formulas without losing decidability? Such a decidable extension could be used to extend guarded programs (as their decidability depends on a translation to guarded fixed point logic) and, consequently, to simulate number restrictions in $\mathcal{DLR}$.

Finally, it would be interesting to define a semantics for the combination of $\mathcal{DLRO}$ with programs (instead of just queries). We conjecture that a semantics as in [27], qualifies for a reduction of a combination of a $\mathcal{DLR}$-like language with a logic program to the unified framework of OASP alone.

---

[8]It does not use nominals.

8. REFERENCES

[1] G. Alsaç and C. Baral. Reasoning in Description Logics using Declarative Logic Programming, 2002.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.

[3] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/, 2004.

[4] K. V. Belleghem, M. Denecker, and D. D. Schreye. A Strong Correspondence between DLs and Open Logic Programming. In *Proc. of ICLP'97*, pages 346–360, 1997.

[5] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Perspectives of Mathematical Logic. Springer, 1997. Second printing (Universitext) 2001.

[6] D. Calvanese, G. De Giacomo, and M. Lenzerini. Conjunctive Query Containment in Description Logics with n-ary Relations. In *Proc. of the 1997 Description Logic Workshop (DL'97)*, pages 5–9, 1997.

[7] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[8] J. de Bruijn, A. Polleres, R. Lara, and D. Fensel. OWL DL vs. OWL Flight: Conceptual Modeling and Reasoning for the Semantic Web. In *Proc. of the International World Wide Web Conference (WWW 2005)*. ACM, 2005.

[9] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: Integrating Datalog and Description Logics. *J. of Intell. and Cooperative Information Systems*, 10:227–252, 1998.

[10] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with DLs for the Semantic Web. In *Proc. of KR 2004*, pages 141–151, 2004.

[11] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-Founded Semantics for Description Logic Programs in the Semantic Web. In *Proc. of RuleML 2004*, number 3323 in LNCS, pages 81–97. Springer, 2004.

[12] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. of ICLP'88*, pages 1070–1080, Cambridge, Massachusetts, 1988. MIT Press.

[13] E. Grädel. Model Checking Games. In *Proceedings of WOLLIC 02*, volume 67 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.

[14] E. Grädel and I. Walukiewicz. Guarded Fixed Point Logic. In *Proc. of LICS '99*, pages 45–54. IEEE Computer Society, 1999.

[15] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proc. of WWW 2003*, pages 48–57, 2003.

[16] S. Heymans, D. Van Nieuwenborgh, and D. Vermeir. Semantic Web Reasoning with Conceptual Logic Programs. In *Proc. of RuleML 2004*, pages 113–127. Springer, 2004.

[17] S. Heymans, D. Van Nieuwenborgh, and D. Vermeir. Guarded Open Answer Set Programming. In *8th International Conference on Logic Programming and Non Monotonic Reasoning (LPNMR 2005)*, number 3662 in LNAI, pages 92–104. Springer, 2005.

[18] S. Heymans, D. Van Nieuwenborgh, and D. Vermeir. Guarded Open Answer Set Programming with Generalized Literals. In *Fourth International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2006)*. Springer, 2005. To Appear.

[19] S. Heymans, D. Van Nieuwenborgh, and D. Vermeir. Nonmonotonic Ontological and Rule-Based Reasoning with Extended Conceptual Logic Programs. In *Proc. of ESWC 2005*, number 3532 in LNCS, pages 392–407. Springer, 2005.

[20] S. Heymans and D. Vermeir. Integrating Description Logics and Answer Set Programming. In *Proc. of PPSWR 2003*, number 2901 in LNCS, pages 146–159. Springer, 2003.

[21] I. Horrocks, P. F. Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule language Combining OWL and RuleML, May 2004.

[22] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. FZI-Report 1-8-11/03, Forschungszentrum Informatik (FZI), 2003.

[23] A. Y. Levy and M. Rousset. CARIN: A Representation Language Combining Horn Rules and Description Logics. In *Proc. of ECAI'96*, pages 323–327, 1996.

[24] V. Lifschitz, D. Pearce, and A. Valverde. Strongly Equivalent Logic Programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

[25] B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. In *Proc. of ISWC 2004*, number 3298 in LNCS, pages 549–563. Springer, 2004.

[26] R. Rosati. Towards Expressive KR Systems Integrating Datalog and Description Logics: Preliminary Report. In *Proc. of DL'99*, pages 160–164, 1999.

[27] R. Rosati. On the Decidability and Complexity of Integrating Ontologies and Rules. *Journal of Web Semantics*, 3(1), 2005.

[28] T. Swift. Deduction in Ontologies via Answer Set Programming. In V. Lifschitz and I. Niemelä, editors, *LPNMR*, volume 2923 of *LNCS*, pages 275–288. Springer, 2004.