# G-Hybrid Knowledge Bases

Stijn Heymans[1], Livia Predoiu[1], Cristina Feier[1], Jos de Bruijn[1], and Davy Van Nieuwenborgh[2⋆]

[1] Digital Enterprise Research Institute (DERI)
University of Innsbruck, Austria
{stijn.heymans,livia.predoiu,cristina.feier,jos.debruijn}@deri.org
[2] Dept. of Computer Science
Vrije Universiteit Brussel, VUB
Pleinlaan 2, B1050 Brussels, Belgium
dvnieuwe@vub.ac.be

**Abstract.** Recently, there has been a lot of interest in the integration of Description Logics and rules on the Semantic Web. We define *g-hybrid knowledge bases* as knowledge bases that consist of a Description Logic knowledge base and a *guarded* logic program, similarly to the $\mathcal{DL}$+*log* knowledge bases from [25]. G-hybrid knowledge bases enable an integration of Description Logics and Logic Programming where, unlike in other approaches, variables in the rules of a guarded program do not need to appear in positive non-DL atoms of the body: DL atoms can act as *guards* as well. Decidability of satisfiability checking of g-hybrid knowledge bases is shown for the particular DL $\mathcal{DLRO}^{-\{\leq\}}$, which is close to, and in some respects more expressive than, OWL DL, by a reduction to guarded programs under an open answer set semantics. Moreover, we show 2-EXPTIME-completeness for satisfiability checking of those $\mathcal{DLRO}^{-\{\leq\}}$ g-hybrid knowledge bases. Finally, we discuss advantages and disadvantages of our approach compared with $\mathcal{DL}$+*log* knowledge bases.

## 1 Introduction

There has been a lot of attention recently in the integration of Description Logics with rules for the Semantic Web [23, 25, 6, 22, 16]. R-hybrid knowledge bases [23], and the extension $\mathcal{DL}$+*log* [25], is an elegant formalism based on combined models for Description Logic knowledge bases and nonmonotonic logic programs. We propose a variant of r-hybrid knowledge bases, called *g-hybrid knowledge bases*, which do not require standard names or a safeness restriction on rules. We show several computational properties by a reduction to guarded open answer set programming [13].

Open answer set programming (OASP) [14, 13] combines the logic programming and first-order logic paradigms. From the logic programming paradigm it inherits a rule-based presentation and a nonmonotonic semantics by means of negation as failure.

⋆ The first four authors were partially supported by the European Commission under projects Knowledge Web (IST-2004-507482) and DIP (FP6-507483) and by the FIT-IT under the project RW$^2$ (FIT-IT 809250). The last author was supported by the Flemish Fund for Scientific Research (FWO-Vlaanderen).

In contrast with usual logic programming semantics, see, e.g., the answer set semantics [8], 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 its rule-based presentation and its 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 [14].

The main challenge for OASP is to control undecidability of satisfiability checking, a challenge it shares with DL-based languages. In [13], a decidable class of programs is identified, so-called *guarded programs*, for which decidability of satisfiability checking is obtained by a translation to guarded fixed point logic [10]. In [12], we show the expressiveness of such guarded programs by simulating a DL with $n$-ary roles and nominals. In particular, we extend the DL $\mathcal{DLR}$ [4] with both *concept nominals* $\{o\}$ and *role nominals* $\{(o_1, \ldots, o_n)\}$, resulting in $\mathcal{DLRO}$. The DL $\mathcal{DLRO}$ with the number restrictions left out yields $\mathcal{DLRO}^{-\{\leq\}}$ and we show in [13] a reduction of satisfiability of concept expressions w.r.t. $\mathcal{DLRO}^{-\{\leq\}}$ knowledge bases to guarded programs.

G-hybrid knowledge bases consist of Description Logic knowledge base and a guarded program. The $\mathcal{DL}$+*log* knowledge bases from [25] are *weakly safe*, i.e., the interaction between the program and the DL knowledge base is limited by imposing that head variables should appear in atoms that cannot be DL atoms (i.e., concepts or roles in the knowledge base). However, in g-hybrid knowledge bases such a restriction does not hold; variables should appear in a *guard* of the rule but this guard can be a DL atom as well. We show decidability of g-hybrid knowledge bases for $\mathcal{DLRO}^{-\{\leq\}}$ DL knowledge bases by a reduction to guarded programs only, as well as provide a 2-EXPTIME complexity characterization of such g-hybrid knowledge bases. $\mathcal{DLRO}^{-\{\leq\}}$ includes a large fragment of $\mathcal{SHOIN}$, the Description Logic underlying OWL DL [15]. Compared with $\mathcal{SHOIN}$, $\mathcal{DLRO}^{-\{\leq\}}$ does not include transitive roles and number restrictions, but does include $n$-ary roles and complex role expressions.

The remainder of the paper starts with an introduction to open answer set programming and Description Logics in Subsections 2.1 and 2.2. Section 3 defines g-hybrid knowledge bases, translates them to guarded programs when the $\mathcal{DLRO}^{-\{\leq\}}$ DL is considered, and provides a complexity characterization for satisfiability checking of these particular g-hybrid knowledge bases. In Section 4, we discuss the relation of g-hybrid knowledge bases with $\mathcal{DL}$+*log* and point to other related work. We conclude and give directions for further research in Section 5.

## 2 Preliminary Definitions: Open Answer Set Programming and Description Logics

In this section, we introduce Open Answer Set Programming and the Description Logic $\mathcal{DLRO}^{-\{\leq\}}$.

### 2.1 Decidable Open Answer Set Programming

We introduce the open answer set semantics from [13], modified as in [12] such that it does not take on a unique name assumption for constants by default. *Constants*, *vari-*

*ables*, *terms*, and *atoms* are defined as usual. A *literal* is an atom $p(\boldsymbol{t})$ or a *naf-atom* $not\ p(\boldsymbol{t})$.[3] The *positive part* of a set of literals $\alpha$ is $\alpha^+ = \{p(\boldsymbol{t}) \mid p(\boldsymbol{t}) \in \alpha\}$ and the *negative part* of $\alpha$ is $\alpha^- = \{p(\boldsymbol{t}) \mid not\ p(\boldsymbol{t}) \in \alpha\}$. 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 $A$ of atoms, $not\ A = \{not\ l \mid l \in A\}$.

A *program* is a countable set of rules $\alpha \leftarrow \beta$, where $\alpha$ and $\beta$ are finite sets of literals, $|\alpha^+| \leq 1$ (but $\alpha^-$ may be of arbitrary size) , and $\forall t, s \cdot t = s \notin \alpha^+$, i.e., $\alpha$ contains at most one positive atom and this atom cannot be an equality atom.[4] The set $\alpha$ is the *head* of the rule and represents a disjunction of literals, while $\beta$ 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 $q(\boldsymbol{t}) \vee not\ q(\boldsymbol{t}) \leftarrow$ for a tuple $\boldsymbol{t}$ of terms; they enable a choice for the inclusion of atoms. We call a predicate $p$ free if there is a free rule $p(\boldsymbol{t}) \vee not\ p(\boldsymbol{t}) \leftarrow$. 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 *pre-interpretation* $U$ for a program $P$ is a pair $(D, \sigma)$ where $D$ is a non-empty *domain* and $\sigma : cts(P) \rightarrow D$ is a function that maps all constants to elements from $D$.[5] 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

$$p(x) \leftarrow f(x, x)$$
$$p(y) \leftarrow f(y, x)$$

Let $\mathcal{B}_P$ be the set of regular atoms that can be defined using the language of the ground program $P$.

An *interpretation* $I$ of a ground $P$ is any subset of $\mathcal{B}_P$. For a ground regular atom $p(\boldsymbol{t})$, we write $I \models p(\boldsymbol{t})$ if $p(\boldsymbol{t}) \in I$; for an equality atom $p(\boldsymbol{t}) \equiv t = s$, we have $I \models p(\boldsymbol{t})$ if $s$ and $t$ are equal terms. We have $I \models not\ p(\boldsymbol{t})$ if $I \not\models p(\boldsymbol{t})$. For a set of ground literals $A$, $I \models A$ if $I \models l$ for every $l \in A$. 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 $I \models \beta$. 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* [8, 20] 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 $P^I$. Note that allowing for negation in the head of rules leads to the loss of the *anti-chain property* which says that no answer set can

[3] We do not allow "classical" negation $\neg$, however, programs with $\neg$ can be reduced to programs without it, see e.g. [21].

[4] The condition $|\alpha^+| \leq 1$ makes the GL-reduct non-disjunctive, ensuring that the *immediate consequence operator* is well-defined, see [13].

[5] In [13], we only use the domain $D$ which is there a non-empty superset of the constants in $P$. This corresponds to a pre-interpretation $(D, \sigma)$ where $\sigma$ is the identity function on $D$.

be a strict subset of another answer set. In the presence of negation in the head answer sets can be subsets of other answer sets. E.g, a rule $a \vee not\ a \leftarrow$ has the answer sets $\emptyset$ and $\{a\}$ . However, we need negation in the head to be able to simulate a first-order behavior for certain predicates, e.g., when simulating Description Logic reasoning.

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 pre-interpretation. An *open interpretation* of a program $P$ is a pair $(U, M)$ where $U$ is a pre-interpretation 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 $\boldsymbol{x} \in D^n$ such that $p(\boldsymbol{x}) \in M$. A program $P$ is satisfiable iff it has an open answer set. Note that satisfiability checking of programs can be easily reduced to satisfiability checking of predicates: $P$ is satisfiable iff $p$ is satisfiable w.r.t. $P \cup \{p(\boldsymbol{X}) \vee not\ p(\boldsymbol{X}) \leftarrow\}$, where $p$ is a new predicate not in $P$ and $\boldsymbol{X}$ is a tuple of variables. In the following, when we speak of satisfiability checking we are referring to satisfiability checking of predicates, unless specified otherwise.

Satisfiability checking w.r.t. the open answer set semantics is undecidable in general. In [13], 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 [10]. 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 $\gamma_b$ 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 [13] – a result that stems from the corresponding complexity in guarded fixed point logic.

We do not have a unique name assumption, i.e., it might be the case that for two distinct $c_1$ and $c_2$, $\sigma(c_1) = \sigma(c_2)$ for a pre-interpretation $(D, \sigma)$.

### 2.2 The Description Logic $\mathcal{DLRO}^{-\{\leq\}}$

The DL $\mathcal{DLR}$ [4, 2] is a DL that supports $n$-ary roles, instead of the usual binary ones. We introduce the extension of $\mathcal{DLR}$ with nominals, called $\mathcal{DLRO}$, as in [12]. 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 $2 \leq n \leq n_{max}$ and $n_{max}$ is a given finite non-negative integer. Role expressions $\mathbf{R}$ and concept expressions $C$ can be formed according to the following syntax rules:

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

where we assume $i$ is between 1 and $n$ in $(\$i/n : C)$, and similarly in $\exists[\$i]\mathbf{R}$ and $\leq k[\$i]\mathbf{R}$ 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 $\mathcal{DLRO}$ 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 $n$):

$$\begin{aligned}
\top_n^{\mathcal{I}} &\subseteq (\Delta^{\mathcal{I}})^n \\
\mathbf{P}^{\mathcal{I}} &\subseteq \top_n^{\mathcal{I}} \\
(\neg \mathbf{R})^{\mathcal{I}} &= \top_n^{\mathcal{I}} \backslash \mathbf{R}^{\mathcal{I}} \\
(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} &= \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}} \\
(\$i/n : C)^{\mathcal{I}} &= \{(d_1, \ldots, d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\} \\
\\
\top_1^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists [\$i] \mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \exists (d_1, \ldots, d_n) \in \mathbf{R}^{\mathcal{I}} \cdot d_i = d\} \\
(\leq k [\$i] \mathbf{R})^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid |\{(d_1, \ldots, d_n) \in \mathbf{R}^{\mathcal{I}} \mid d_i = d\}| \leq k\} \\
\{o\}^{\mathcal{I}} &= \{o^{\mathcal{I}}\} \subseteq \Delta^{\mathcal{I}} \\
\{(o_1, \ldots, o_n)\}^{\mathcal{I}} &= \{(o_1^{\mathcal{I}}, \ldots, o_n^{\mathcal{I}})\}
\end{aligned}$$

Note that in $\mathcal{DLRO}$ the negation of role expressions is defined w.r.t. $\top_n^{\mathcal{I}}$ instead of $(\Delta^{\mathcal{I}})^n$. A $\mathcal{DLRO}$ 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, in which case we call the knowledge base *satisfiable*. A concept expression $C$ is satisfiable w.r.t. a knowledge base $\Sigma$ if there is a model $\mathcal{I}$ of $\Sigma$ such that $C^{\mathcal{I}} \neq \emptyset$.

Note that for every interpretation $\mathcal{I}$, one has

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

such that we will restrict ourselves in the remainder of the paper to nominals of the form $\{o\}$ only.

We denote the fragment of $\mathcal{DLRO}$ without the number restriction $\leq k[\$i]\mathbf{R}$ as $\mathcal{DLRO}^{-\{\leq\}}$.

## 3 G-hybrid Knowledge Bases

G-hybrid knowledge bases are a variant of r-hybrid knowledge bases [23], based on guarded programs, without the standard names assumption. Given a particular Description Logic $\mathcal{DL}$, we define *g-hybrid knowledge bases* as pairs consisting of on the one hand a $\mathcal{DL}$ knowledge base and on the other hand a guarded program (GP).

**Definition 1.** *Given Description Logic $\mathcal{DL}$, a* g-hybrid knowledge base *is a pair $(\Sigma, P)$ where $\Sigma$ is a $\mathcal{DL}$ knowledge base and $P$ is a guarded program.*

Note that in the above definition there are no conditions on the occurrence of predicates, however, by definition, we call the atoms and literals in $P$ that have underlying predicates that correspond to concept names or role names in the DL knowledge base, *DL atoms* and *DL literals* respectively. Variables in rules are not required to appear in positive non-DL atoms as is the case in, e.g., the $\mathcal{DL}{+}log$ knowledge bases in [25], the r-hybrid knowledge bases in [23], or the DL-safe rules in [22]. DL-atoms can appear in the head of rules, thus allowing for a bi-directional flow of information between the DL knowledge base and the program.

*Example 1.* Consider the $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base $\Sigma$ where $socialDrinker$ is a concept, $drinks$ is a ternary role such that, intuitively, $(x, y, z)$ is in the interpretation of $drinks$ if a person $x$ drinks with person $y$ some $z$:

$$socialDrinker \sqsubseteq \exists[\$1](drinks \sqcap (\$3/3 : \{wine\})) \ .$$

The knowledge base indicates that social drinkers drink wine with someone. Consider a GP $P$ that indicates that someone has an increased risk of alcoholism if the person is a social drinker and knows someone from the association of Alcoholics Anonymous (AA). Furthermore, we state that *john* is a social drinker and knows *michael* from AA.

$$\begin{aligned} problematic(X) &\leftarrow socialDrinker(X), knowsFromAA(X, Y) \\ knowsFromAA(john, michael) &\leftarrow \\ socialDrinker(john) &\leftarrow \end{aligned}$$

Together $\Sigma$ and $P$ form a g-hybrid knowledge base. The literals $socialDrinker(X)$ and $socialDrinker(john)$ are DL atoms where the latter appears in the head of a rule in $P$. The literal *knowsFromAA(X,Y)* appears only in the program $P$ (and is thus not a DL atom).

We define the semantics of g-hybrid knowledge bases $(\Sigma, P)$ using interpretations $(U, \mathcal{I}, M)$. Given a DL interpretation $(D, \mathcal{I})$ and a ground program $P$, define $\Pi(P, \mathcal{I})$ as the *projection* of $P$ with respect to $\mathcal{I}$ obtained as follows: for every rule $r$ in $P$,

- if there exists a DL literal in the head of the form
  - $A(\boldsymbol{t})$ with $\boldsymbol{t} \in A^{\mathcal{I}}$, or
  - $not\ A(\boldsymbol{t})$ with $\boldsymbol{t} \notin A^{\mathcal{I}}$,

  then delete $r$,
- if there exists a DL literal in the body of the form
  - $A(\boldsymbol{t})$ with $\boldsymbol{t} \notin A^{\mathcal{I}}$, or
  - $not\ A(\boldsymbol{t})$ with $\boldsymbol{t} \in A^{\mathcal{I}}$, or

  then delete $r$,
- otherwise, delete all DL literals from $r$.

Intuitively, the projection of a ground program transforms this grounded program by removing rules and DL literals consistently with $\mathcal{I}$; conceptually this is similar to the GL-reduct which removes the rules and negative literals consistently with an interpretation.

**Definition 2.** *Let $(\Sigma, P)$ be a g-hybrid knowledge base. Then $(U, \mathcal{I}, M)$ is an interpretation of $(\Sigma, P)$ iff*

- *$U = (D, \sigma)$ is a pre-interpretation for $P$,*
- *$(D, \mathcal{I})$ is an interpretation of $\Sigma$,*
- *$M$ is an interpretation of $\Pi(P_U, \mathcal{I})$, and*
- *$b^{\mathcal{I}} = \sigma(b)$ for every constant symbol $b$ appearing both in $\Sigma$ and in $P$,*

For $(U = (D, \sigma), \mathcal{I}, M)$ to be a model of a g-hybrid knowledge base, we require that $(D, \mathcal{I})$ should be a model of the Description Logic knowledge base and that $M$ should be an answer set of the projection of the grounding of the program with $U$.

**Definition 3.** *An interpretation $(U, \mathcal{I}, M)$ with $U = (D, \sigma)$ is then a* model *of $(\Sigma, P)$ iff*

1. *$(D, \mathcal{I})$ is a model of $\Sigma$, and*
2. *$M$ is an answer set of $\Pi(P_U, \mathcal{I})$.*

*For $p$ a concept expression from $\Sigma$ or a predicate from $P$, we have that $p$ is* satisfiable *w.r.t $(\Sigma, P)$ if there is a model $(U, \mathcal{I}, M)$ such that $p^{\mathcal{I}} \neq \emptyset$ or $p(\boldsymbol{x}) \in M$ for some $\boldsymbol{x}$ from $D$, respectively.*

*Example 2.* Consider the g-hybrid knowledge base in Example 1. Take $U = (D, \sigma)$ with $D = \{john, michael, wine, x\}$ and $\sigma$ the identity function on the constant symbols in $(\Sigma, P)$. Furthermore, define $\cdot^{\mathcal{I}}$ as follows:

- $socialDrinker^{\mathcal{I}} = \{john\}$,
- $drinks^{\mathcal{I}} = \{(john, x, wine)\}$,
- $wine^{\mathcal{I}} = wine$.

and $M \equiv \{knowsfromAA(john, michael), problematic(john)\}$. Then $(U, \mathcal{I}, M)$ is a model of this g-hybrid knowledge base. Note that the projection of the program will no longer contain the rule $socialDrinker(john) \leftarrow$ .

We can translate the g-hybrid knowledge base from Example 1 to a GP such that the knowledge base is satisfiable iff the logic program is satisfiable. The axiom

$$socialDrinker \sqsubseteq \exists[\$1](drinks \sqcap (\$3/3 : \{wine\})) \ .$$

is translated to a constraint:

$$\leftarrow socialDrinker(X), not\ (\exists[\$1](drinks \sqcap (\$3/3 : \{wine\})))(X)$$

Thus, the concept expressions on either side of the $\sqsubseteq$ symbol in an axiom are associated with a new unary predicate name. For convenience, we denote the predicates like the corresponding concept expressions. The constraint simulates the behavior of the $\mathcal{DLR}\mathcal{O}^{-\{\leq\}}$ axiom. If the left-hand side of the axiom holds and the right-hand side does not hold, we have a contradiction.

It remains to define those newly introduced predicates according to the DL semantics. First, all the concept and role names occurring in the axiom above need to be

defined as free predicates, in order to simulate the first-order semantics of concept and role names in DLs. Intuitively, in DLs a tuple is in the extension of a concept or role or not; this behavior can be captured exactly by free predicates:

$$socialDrinker(X) \vee not\ socialDrinker(X) \leftarrow$$
$$drinks(X, Y, Z) \vee not\ drinks(X, Y, Z) \leftarrow$$

Note that concept names are translated to unary free predicates while $n$-ary role names are translated to $n$-ary free predicates.

The definition of the truth symbols $\top_1$ and $\top_3$ that are implicitly present in our $\mathcal{DLRO}^{-\{\leq\}}$ axiom (since the axiom contains a concept name and a ternary role) are defined as free predicates as well. Note that we do not need a predicate for $\top_2$ since the axiom does not contain binary predicates.

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

We ensure that for the ternary $\mathcal{DLRO}^{-\{\leq\}}$ role $drinks$, $drinks^{\mathcal{I}} \subseteq \top_3^{\mathcal{I}}$ holds by adding the constraint:

$$\leftarrow drinks(X, Y, Z), not\ \top_3(X, Y, Z)$$

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 such that the latter rule is a guarded rule when regarded as the equivalent rule $\leftarrow not\ \top_1(X), X = X$. Note that we can allow for such an equality guard without affecting decidability as decidability for guarded programs was shown in [13] by a translation to guarded fixed point logic where one allows for guards $X = X$ as well [9].

We define the nominal $\{wine\}$ by the rule

$$\{wine\}(wine) \leftarrow$$

Intuitively, since this rule will be the only rule with the predicate $\{wine\}$ in the head, every open answer set of the translated program will only contain $\{wine\}(x)$ with $\sigma(wine) = x$ if and only if the corresponding interpretation $\{wine\}^{\mathcal{I}} = \{x\}$ for $wine^{\mathcal{I}} = x$.

The $\mathcal{DLRO}^{-\{\leq\}}$ role expression $(\$3/3 : \{wine\})$ indicates the ternary tuples for which the third argument belongs to the extension of $wine$, which translates to the following rule:

$$(\$3/3 : \{wine\})(X, Y, Z) \leftarrow \top_3(X, Y, Z), \{wine\}(Z)$$

Note that the above rule is guarded by the $\top_3$ literal.

Finally, the concept expression $(drinks \sqcap (\$3/3 : \{wine\}))$ can be represented by the following rule:

$$(drinks \sqcap (\$3/3 : \{wine\}))(X, Y, Z) \leftarrow drinks(X, Y, Z), \\ (\$3/3 : \{wine\})(X, Y, Z)$$

The translation thus translates the DL constructor $\sqcap$ as conjunction in the body of a rule.

The $\mathcal{DLR}\mathcal{O}^{-\{\leq\}}$ role $\exists[\$1](drinks \sqcap (\$3/3 : \{wine\}))$ can be represented by the following rule:

$$(\exists[\$1](drinks \sqcap (\$3/3 : \{wine\})))(X) \leftarrow (drinks \sqcap (\$3/3 : \{wine\}))(X, Y, Z)$$

Indeed, the elements that belong to the extension of $\exists[\$1](drinks \sqcap (\$3/3 : \{wine\}))$ are exactly those that are connected with the role $(\$3/3 : \{wine\})$ as specified in the rule.

This concludes the translation of the DL knowledge base part of the g-hybrid knowledge base in Example 1. The program part can be considered as is, since, by definition of g-hybrid knowledge bases, this is already a GP.

We define the formal translation from g-hybrid satisfiability checking to satisfiability checking w.r.t. programs using the notion of *closure*. Define the *closure* $clos(\Sigma)$ of a $\mathcal{DLR}\mathcal{O}^{-\{\leq\}}$ knowledge base $\Sigma$ as the smallest set satisfying the following conditions:

- $\top_1 \in clos(\Sigma)$,
- for each $C \sqsubseteq D$ an axiom in $\Sigma$ (role or terminological), $\{C, D\} \subseteq clos(\Sigma)$,
- for every $D$ in $clos(\Sigma)$, $clos(\Sigma)$ should contain every subformula that is a concept expression or a role expression,
- if $clos(\Sigma)$ contains $n$-ary relation names, it must contain $\top_n$.

Formally, we define $\Phi(\Sigma)$ for a $\mathcal{DLR}\mathcal{O}^{-\{\leq\}}$ knowledge base $\Sigma$ to be the following program:

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

$$\leftarrow C(X), not\ D(X) \tag{1}$$

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

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

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

$$\top_n(X_1, \ldots, X_n) \vee not\ \top_n(X_1, \ldots, X_n) \leftarrow \tag{3}$$

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

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

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

$$\leftarrow not\ \top_1(X) \tag{5}$$

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

– Next, we distinguish between the types of concept and role expressions that appear in $clos(\Sigma)$. For $D \in clos(\Sigma)$:
  - if $D$ is a concept nominal $\{o\}$, add

$$D(o) \leftarrow \tag{6}$$

    This will ensure that $\{o\}(x)$ holds in an open answer set iff $x = \sigma(o) = o^{\mathcal{I}}$ for an interpretation of $(\Sigma, P)$.
  - if $D$ is a concept name, add

$$D(X) \vee not\ D(X) \leftarrow \tag{7}$$

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

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

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

$$D(X) \leftarrow not\ E(X) \tag{9}$$

    Note that we can again assume that such a rule is guarded by $X = X$.
  - if $D = \neg\mathbf{R}$ for an $n$-ary role expression $\mathbf{R}$, add

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

    Note that if negation was defined w.r.t. to $D^n$ 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

$$D(X) \leftarrow E(X), F(X) \tag{11}$$

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

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

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

$$D(X_1, \ldots, X_i, \ldots, X_n) \leftarrow \top_n(X_1, \ldots, X_i, \ldots, X_n), C(X_i) \tag{13}$$

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

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

We now show that this translation preserves satisfiability.

**Theorem 1.** *Let $(\Sigma, P)$ be a g-hybrid knowledge base where $\Sigma$ is a $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base. Then, a predicate or concept expression $p$ is satisfiable w.r.t. $(\Sigma, P)$ iff $p$ is satisfiable w.r.t. $\Phi(\Sigma) \cup P$.*

*Proof.* ($\Rightarrow$) Assume $p$ is satisfiable w.r.t. $(\Sigma, P)$, i.e., there exists a model $(U, \mathcal{I}, M)$ of $(\Sigma, P)$ where $U$ is the pre-interpretation $(D, \sigma)$ that gives $p$ a non-empty extension. Construct then the open interpretation $(V, N)$ of $(\Sigma, P)$ such that $V = (D, \sigma')$ with $\sigma' : cts(\Phi(\Sigma) \cup P) \rightarrow D$ defined such that $\sigma'(x) = \sigma(x)$ for a constant symbol $x$ from $P$ and $\sigma'(x) = x^{\mathcal{I}}$ for a constant symbol from $\Sigma$. Note that $\sigma'$ is well-defined since for constant symbols $x$ that are in both $\Sigma$ and $P$, we have that $\sigma(x) = x^{\mathcal{I}}$. The set $N$ is defined as follows:

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

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

It is easy to verify that $(V, N)$ is an open answer set of $\Phi(\Sigma) \cup P$ that satisfies $p$.

($\Leftarrow$) Assume $(V, N)$ is an open answer set of $\Phi(\Sigma) \cup P$ with $V = (D, \sigma')$ such that $p$ is satisfied. Define a tuple $(U, \mathcal{I}, N)$, with

- $U \equiv (D, \sigma)$ where $\sigma : cts(P) \rightarrow D$ with $\sigma(x) \equiv \sigma'(x)$ (note that this is possible since $cts(P) \subseteq cts(\Phi(\Sigma) \cup P)$). $U$ is then a pre-interpretation for $P$.
- An interpretation function $\mathcal{I}$ defined such that $A^{\mathcal{I}} \equiv \{x \mid A(x) \in N\}$ for concept names $A$, $\mathbf{R}^{\mathcal{I}} \equiv \{(x_1, \ldots, x_n) \mid \mathbf{R}(x_1, \ldots, x_n) \in N\}$ for $n$-ary role names $\mathbf{R}$ and $o^{\mathcal{I}} = \sigma'(o)$, for $o$ a constant symbol in $\Sigma$ (note that $\sigma'$ is indeed defined on $o$). $(D, \mathcal{I})$ is then an interpretation of $\Sigma$.
- $M \equiv N \setminus \{p(\boldsymbol{x}) \mid p \in clos(\Sigma)\}$, such that $M$ is an interpretation of $\Pi(P_U, \mathcal{I})$.

Moreover, for every constant symbol $b$ appearing both in $\Sigma$ and in $P$, $b^{\mathcal{I}} = \sigma(b)$, making $(U, \mathcal{I}, M)$ an interpretation of $(\Sigma, P)$.

It is easy to verify that $(U, \mathcal{I}, M)$ is a model of $(\Sigma, P)$ that satisfies $p$. □

**Theorem 2.** *Let $(\Sigma, P)$ be a g-hybrid knowledge base where $\Sigma$ is a $\mathcal{DLR}\mathcal{O}^{-\{\leq\}}$ knowledge base. Then, $\Phi(\Sigma) \cup P$ is a GP, with a size that is polynomial in the size of $(\Sigma, P)$.*

*Proof.* Observing the rules that originate from $\Sigma$, it is clear that they are guarded. Furthermore, the program $P$ itself is a GP such that $\Phi(\Sigma) \cup P$ is as well.

The size of $clos(\Sigma)$ is of the order $n \log n$ where $n$ is the size of $\Sigma$. Indeed, intuitively, given that the size of an expression is $n$, we have that the size of the set of its subexpressions is at most the size of a tree with depth $\log n$ where the size of the subexpressions at a certain level of the tree is at most $n$. The size of the GP $\Phi(\Sigma)$ is polynomial in the size of $clos(\Sigma)$. However, note that we assume here that the size of $\Sigma$ 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 $\Sigma$. If we were to add a relation name $\mathbf{R}$ with arity $2^n$, where $n$ is the maximal arity of relation names in $C$ and $\Sigma$, the size of $\Sigma$ would increase linearly, but the size of $\Phi(\Sigma) \cup P$ would increase exponentially: one needs to add, e.g., rules

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

which introduce an exponential number of arguments while the size of the role $\mathbf{R}$ does not depend on its arity. □

Note that in g-hybrid knowledge bases, we consider 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 GPs. From Theorems 1 and 2 follows:

**Corollary 1.** *Satisfiability checking w.r.t. g-hybrid knowledge bases where the DL part is a $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base can be polynomially reduced to satisfiability checking w.r.t. GPs.*

Since satisfiability checking w.r.t. GPs is 2-EXPTIME-complete [13], we have the same 2-EXPTIME characterization for g-hybrid knowledge bases. We first make explicit a corollary of Theorem 1.

**Corollary 2.** *Let $P$ be a GP. Then, $p$ is satisfiable w.r.t. $P$ iff $p$ is satisfiable w.r.t. $(\emptyset, P)$.*

**Theorem 3.** *Satisfiability checking w.r.t. g-hybrid knowledge bases where the DL part is a $\mathcal{DLRO}^{-\{\leq\}}$ knowledge base is 2-EXPTIME-complete.*

*Proof.* Membership in 2-EXPTIME follows from Corollary 1. Hardness follows from 2-EXPTIME-hardness of satisfiability checking w.r.t. GPs and the reduction to satisfiability checking in Corollary 2. □

## 4 Relation with $\mathcal{DL}$+*log* and other Related Work

In [25], so-called $\mathcal{DL}$+*log* knowledge bases combine a Description Logic knowledge base with a *weakly-safe* disjunctive logic program. Formally, for a particular Description Logic $\mathcal{DL}$, a $\mathcal{DL}$+*log knowledge base* is a pair $(\Sigma, P)$ where $\Sigma$ is a $\mathcal{DL}$ knowledge base consisting of a *TBox* (a set of terminological axioms) and an *ABox* (a set of *assertional axioms*), and $P$ contains rules $\alpha \leftarrow \beta$ such that for every rule $r : \alpha \leftarrow \beta \in P$:

- $\alpha^- = \emptyset$,
- $\beta^-$ does not contain DL atoms (call this *DL-positiveness*),
- each variable in $r$ occurs in an atom from $\beta^+$ (*Datalog safeness*), and
- each head variable in $r$ occurs in a non-DL atom from $\beta^+$ (*weak safeness*).

The semantics for $\mathcal{DL}$+*log* is the same as it is for g-hybrid knowledge bases[6], with some exceptions:

[6] Strictly speaking, we did not define answer sets of disjunctive programs, however, the definitions of Subsection 2.1 can serve for disjunctive programs without modification. Also, we did not consider ABoxes in our definition of DLs in Subsection 2.2. However, the extension of the semantics of DL knowledge bases with ABoxes is straightforward.

– We do not have a *standard name assumption* such as [25] has, which basically assumes every interpretation is over the same infinitely countable number of constants. Neither do we have the implied *unique name assumption*, making the semantics for g-hybrid knowledge bases more in line with current Semantic Web standards such as OWL [3] where neither the standard names assumption nor the unique names assumption holds. Note that Rosati also presented a version of hybrid knowledge bases which does not adhere to the unique name assumption in an earlier work [24]. However, the grounding of the program part is with the constant symbols explictly appearing in the program or DL part only, which yields a less tight integration of the program and the DL part than in [25] or in g-hybrid knowledge bases.
– Furthermore, we defined an interpretation as a triple $(U, \mathcal{I}, M)$ instead of a pair $(U, \mathcal{I}')$ where $\mathcal{I}' = \mathcal{I} \cup M$; this is, however, equivalent to [25].

We balance the key differences of the two approaches:

– In [25] the head of a rule is of the form $p_1(\boldsymbol{X}_1) \vee \ldots \vee p_n(\boldsymbol{X}_n)$ with $n$ possibly 0, i.e., the requirement $|\alpha^+| \leq 1$ does not hold as it does for our programs. On the other hand, this implies that $|\alpha^-| = 0$ in [25], while there is no such restriction in our case.
– Instead of Datalog safeness we have *guardedness*, i.e., while with Datalog safeness every variable in the rule should appear in some positive atom of the body of the rule, guardedness requires that there is a positive atom that contains every variable in the rule. E.g., $a(X) \leftarrow b(X), c(Y)$ is Datalog safe since $X$ appears in $b(X)$ and $Y$ appears in $c(Y)$ but it is not guarded since there is no atom that contains both $X$ and $Y$ in its arguments. Note that we could easily extend the approach taken in this paper to *loosely guarded programs* which require that every two variables in the rule should appear together in a positive atom, however, this would still be less expressive than Datalog safeness.
– We do not have the requirement for weak safeness, i.e., head variables do not need to appear positively in a non-DL atom. The guardedness may be provided by a DL atom.

  *Example 3*. Example 1 contains the rule

  $$problematic(X) \leftarrow socialDrinker(X), knowsFromAA(X, Y)$$

  This allows to deduce that $X$ might be a problem case even if $X$ knows someone from the AA but is not drinking with that person, indeed, as illustrated by the example model in Example 1, $john$ is drinking wine with some anonymous $x$ and knows $michael$ from the AA. More correct would be the rule

  $$problematic(X, Z) \leftarrow drinks(X, Y, Z), knowsFromAA(X, Y)$$

  where we explicitly say that $X$ and $Y$ in the $drink$ and $knowsFromAA$ relation should be the same and we extend the $problematic$ predicate with the kind of drink that $X$ has a problem with. Then, the head variable $Z$ is guarded by the DL atom $drinks$ and the rule is thus not weakly-safe but it is guarded nonetheless. Thus, the resulting knowledge base is not a $\mathcal{DL}$+*log* knowledge base but is a g-hybrid knowledge base.

– We do not have the requirement for DL-positiveness, i.e., DL atoms may appear negated in the body of rules (and also in the heads of rules). However, one could allow this in $\mathcal{DL}{+}log$ knowledge bases as well, since $not\ A(\boldsymbol{X})$ in the body of the rule has the same effect as $A(\boldsymbol{X})$ in the head, where the latter is allowed in [25]. Vice versa, we can also loosen our restriction on the occurrence of positive atoms in the head (which allows at most one positive atom in the head), to allow for an arbitrary number of positive DL atoms in the head (but still keep the number of positive non-DL atoms limited to one). E.g., a rule $p(X) \vee A(X) \leftarrow \beta$, where $A(X)$ is a DL atom, is not a valid rule in the programs we considered since the head contains more than one positive atom. However, we can always rewrite such a rule as the rule $p(X) \leftarrow \beta, not\ A(X)$, which contains at most one positive atom in the head.
  Arguably, DL atoms should not be allowed to occur negatively, because DL predicates are interpreted classically and thus the negation in front of the DL atom is not nonmonotonic. However, Datalog predicates which depend on DL predicates are also (partially) interpreted classically.
– We do not take into account ABoxes in the DL knowledge base like [25] does. However, the DL we consider includes nominals such that one can simulate the ABox using terminological axioms. Moreover, even if the DL does not include nominals the ABox can be written as ground facts in a program and ground facts are always guarded.
– Decidability for satisfiability checking[7] of $\mathcal{DL}{+}log$ knowledge bases in [25] is guaranteed if decidability of the conjunctive query containment problem is guaranteed for the DL at hand. However, we relied for showing decidability on a translation of DLs to guarded programs, and, as explained in the previous section, e.g., DLs with number restrictions cannot be translated to them.

[18] and [26] simulate reasoning in DLs with a LP formalism by using an intermediate translation to first-order clauses. In [18], $\mathcal{SHIQ}$ knowledge bases are reduced to first-order formulas, on which basic superposition calculus is then applied.

[26] translates $\mathcal{ALCQI}$ concept expressions to first-order formulas, grounds them with a finite number of constants, and transforms the result to a logic program. One can use a finite number of constants by the finite model property of $\mathcal{ALCQI}$; in the presence of terminological axioms this is no longer possible since the finite model property is lost.

In [19], the DL $\mathcal{ALCNR}$ ($\mathcal{R}$ stands for role intersection) is extended with Horn clauses $q(\boldsymbol{Y}) \leftarrow p_1(\boldsymbol{X}_1), \ldots, p_n(\boldsymbol{X}_n)$ where the variables in $\boldsymbol{Y}$ must appear in $\boldsymbol{X}_1 \cup \ldots \cup \boldsymbol{X}_n$; $p_1, \ldots, p_n$ are either concept names, role names, or ordinary predicates not appearing in the DL part, and $q$ is an ordinary predicate. There is no safeness in the sense that every variable must appear in a non-DL atom (i.e., with an ordinary predicate), as it is in, e.g., [22]. The semantics is as in [22]: extended interpretations that satisfy both the DL and clauses part (as FOL formulas). Query answering is undecidable if recursive Horn clauses are allowed, but decidability can be regained by restricting the DL part or

[7] [25] considers satisfiability checking of knowledge bases instead of predicate satisfiability checking as we do, however, the former can easily be reduced to the latter.

by enforcing that the clauses are role safe (each variable in a role atom $R(X, Y)$ for a role $R$ must appear in a non-DL atom). Note that the latter restriction is less strict than the DL-safeness of [22], where also variables in concept atoms $A(X)$ need to appear in non-DL atoms. On the other hand, [22] allows for the more expressive DL $\mathcal{SHOIN}(\mathbf{D})$, and the head predicates may be DL atoms as well. In relation with our work: we needed the guardedness and not just role safeness as in [19].

An $\mathcal{AL}$-log [5] system consists of two subsystems: an $\mathcal{ALC}$ knowledge base and a set of Horn clauses of the above form, where variables in the head must appear in the body, only concept names besides ordinary predicates are allowed in the body (thus no role names), and there is a safeness condition as in [22] saying that every variable appears in a non-DL atom.

In [6, 7] *Description Logic programs* are introduced; atoms in the program component may be *dl-atoms* such that one can query the knowledge in the DL part and each query can also provide the DL with information that the rule part deduced, yielding a bi-directional flow of information.

Finally, SWRL [17] is a *Semantic Web Rule Language* and extends the syntax and semantics of OWL DL (i.e., $\mathcal{SHOIN}(\mathbf{D})$) with unary/binary Datalog RuleML [1], i.e., Horn-like rules. This extension is undecidable [16].

## 5 Conclusions and Directions for Further Research

We defined g-hybrid knowledge bases which combine Description Logic (DL) knowledge bases with guarded programs. In particular, we combined knowledge bases of the DL $\mathcal{DLRO}^{-\{\leq\}}$, which is close to OWL DL, with guarded programs and showed decidability of this framework by a reduction to guarded programs under the open answer set semantics [13]. We discussed the relation with $\mathcal{DL}$+*log* knowledge bases: g-hybrid knowledge bases overcome some of the limitations of $\mathcal{DL}$+*log*, such as the unique name assumption, the requirement for DL-positiveness, Datalog safeness, and weak DL-safeness, but introduces the requirement of guardedness. At present, a significant disadvantage of our approach is the lack of support for DLs with number restrictions which is inherent to the use of guarded programs as our decidability vehicle. A solution for this would be to consider other types of programs, such as *conceptual logic programs* [11]. This would allow for the definition of an hybrid knowledge base $(\Sigma, P)$ where $\Sigma$ is a $\mathcal{SHIQ}$ knowledge base and $P$ is a conceptual logic program since $\mathcal{SHIQ}$ knowledge bases can be translated to conceptual logic programs.

At present, there is no implemented system for open answer set programming available; this is part of future research.

## References


1. The Rule Markup Initiative. http://www.ruleml.org.
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. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. E. Grädel, C. Hirsch, and M. Otto. Back and Forth Between Guarded and Modal Logics. *ACM Transactions on Computational Logic*, 3:418–463, 2002.
10. E. Grädel and I. Walukiewicz. Guarded Fixed Point Logic. In *Proc. of LICS '99*, pages 45–54. IEEE Computer Society, 1999.
11. S. Heymans, D. Van Nieuwenborgh, and D. Vermeir. Conceptual logic programs. *Annals of Mathematics and Artificial Intelligence (Special Issue on Answer Set Programming)*, 2006.
12. S. Heymans, D. Van Nieuwenborgh, D. Fensel, and D. Vermeir. Reasoning with the Description Logic $\mathcal{DLRO}^{-\{\leq\}}$ using Bound Guarded Programs. In *Proc. of Reasoning on the Web workshop (RoW 2006)*, Edinburgh, UK, 2006.
13. S. Heymans, D. Van Nieuwenborgh, and D. Vermeir. Guarded Open Answer Set Programming. In *LPNMR 2005*, number 3662 in LNAI, pages 92–104. Springer, 2005.
14. 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.
15. I. Horrocks and P. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfiability. *J. of Web Semantics*, 2004. To Appear.
16. I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proc. of WWW 2004*. ACM, 2004.
17. 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.
18. 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.
19. 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.
20. V. Lifschitz. Answer Set Programming and Plan Generation. *AI*, 138(1-2):39–54, 2002.
21. V. Lifschitz, D. Pearce, and A. Valverde. Strongly Equivalent Logic Programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.
22. 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.
23. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Web Semantics*, 3(1):41–60, 2005.
24. R. Rosati. Semantic and computational advantages of the safe integration of ontologies and rules. In *Proc. of the Third International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2005)*, volume 3703 of *LNCS*, pages 50–64. Springer, 2005.
25. R. Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In *Proc. of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 68–78. AAAI Press, 2006.
26. T. Swift. Deduction in Ontologies via Answer Set Programming. In Vladimir Lifschitz and Ilkka Niemelä, editors, *LPNMR*, volume 2923 of *LNCS*, pages 275–288. Springer, 2004.