TLP **8** (3): 411–429, 2008.
doi:10.1017/S1471068407003201

Guarded hybrid knowledge bases[1,2]

STIJN HEYMANS
Digital Enterprise Research Institute, University of Innsbruck, Technikerstrasse 21a, Innsbruck, Austria
(*e-mail:* `stijn.heymans@deri.at`)

JOS DE BRUIJN
Faculty of Computer Science, Free University of Bozen-Bolzano, I-39100 Bozen-Bolzano, Italy
(*e-mail:* `debruijn@inf.unibz.it`)

LIVIA PREDOIU
Institute of Computer Science, University of Mannheim, A5, 6 68159 Mannheim, Germany
(*e-mail:* `livia@informatik.uni-mannheim.de`)

CRISTINA FEIER
Digital Enterprise Research Institute, University of Innsbruck, Technikerstrasse 21a, Innsbruck, Austria
(*e-mail:* `cristina.feier@deri.at`)

DAVY VAN NIEWENBORGH
Department of Computer Science, Vrije Universiteit Brussel, VUB, Pleinlaan 2, B1050 Brussels, Belgium
(*e-mail:* `dvnieuwe@vub.ac.be`)

submitted 20 June 2006; revised 3 January 2007; accepted 18 October 2007

Abstract

Recently, there has been a lot of interest in the integration of Description Logics (DL) and rules on the Semantic Web. We define *guarded hybrid knowledge bases* (or *g-hybrid knowledge bases*) as knowledge bases that consist of a Description Logic knowledge base and a *guarded* logic program, similar to the $\mathcal{DL}+log$ knowledge bases from Rosati (*In Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning*, AAAI Press, Menlo Park, CA, 2006, pp. 68–78.). 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, i.e., 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 OWL DL, by a reduction to guarded programs under the open answer set semantics. Moreover, we show 2-EXPTIME-completeness for satisfiability checking of such g-hybrid knowledge bases.

---

[1] A preliminary version of this paper presented at the proceedings of the *ICLP'06 Workshop on Applications of Logic Programming in the Semantic Web and Semantic Web Services (ALPSWS2006)*, Seattle, WA, USA, August 16, 2006, pp. 39–54.

[2] The work has funded by the European Commission under the projects ASG, DIP, enIRaF, InfraWebs, Knowledge Web, Musing, Salero, SEKT, SEEMP, SemanticGOV, Super, SWING and TripCom; by Science Foundation Ireland under the DERI-Lion grant no.SFI/02/CE1/I13; by the FFG (Österreichische Forschungsfrderungsgeselleschaft mbH) under the projects Grisino, RW^2, SemNetMan, SEnSE, TSC and OnTourism. Davy Van Nieuwenborgh was supported by the Flemish Fund for Scientific Research (FWO-Vlaanderen).

Finally, we discuss advantages and disadvantages of our approach compared with $\mathscr{DL}+log$ knowledge bases.

KEYWORDS: g-hybrid knowledge bases, open answer set programming, guarded logic programs, description logics

---

1 Introduction

The integration of Description Logics with rules has recently received a lot of attention in the context of the Semantic Web (Eiter *et al.* 2004; Horrocks and Patel-Schneider 2004b; Motik *et al.* 2004; Rosati 2005a, 2006; de Bruijn *et al.* 2007; Motik and Rosati 2007). r-Hybrid knowledge bases (Rosati 2005a), and its extension $\mathscr{DL}+log$ (Rosati 2006), are 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 special safeness restriction on rules, but instead require the program to be *guarded*. We show several computational properties by a reduction to guarded open answer set programming (Heymans *et al.* 2005a, 2006c).

Open answer set programming (OASP) (Heymans *et al.* 2005a, 2006c) 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. In contrast with usual logic programming semantics, such as the answer set semantics (Gelfond and Lifschitz 1988), 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) (Baader *et al.* 2003) and make OASP a viable candidate for conceptual reasoning. Owing 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 Heymans *et al.* (2005b).

A major challenge for OASP is to control undecidability of satisfiability checking, a challenge it shares with DL-based languages. In Heymans *et al.* (2005a, 2006c), we identify a decidable class of programs, the so-called *guarded programs*, for which decidability of satisfiability checking is obtained by a translation to guarded fixed point logic (Grädel and Walukiewicz 1999). In Heymans *et al.* (2006a), we show the expressiveness of such guarded programs by simulating a DL with n-ary roles and nominals. In particular, we extend the DL $\mathscr{DLR}$ (Calvanese *et al.* 1997) with both *concept nominals* $\{o\}$ and *role nominals* $\{(o_1, \ldots, o_n)\}$, resulting in $\mathscr{DLRO}$. We denote the DL $\mathscr{DLRO}$ without number restrictions as $\mathscr{DLRO}^{-\{\leqslant\}}$. Satisfiability checking of concept expressions w.r.t. $\mathscr{DLRO}^{-\{\leqslant\}}$ knowledge bases can be reduced to checking satisfiability of guarded programs (Heymans *et al.* 2006c).

A g-hybrid knowledge base consists of a Description Logic knowledge base and a guarded program. The $\mathscr{DL}+log$ knowledge bases from Rosati (2006) are *weakly safe*, which means that the interaction between the program and the DL knowledge base is restricted by requiring that variables which appear in non-DL atoms, appear in positive non-DL atoms in the body, where DL atoms are atoms involving a concept or role symbol from the DL knowledge base. g-Hybrid knowledge bases do not require such a restriction; instead, variables must appear in a *guard* of the rule, but this guard can be a DL atom as well. In this paper, we show decidability of g-hybrid knowledge bases for $\mathscr{DLRO}^{-\{\leqslant\}}$ knowledge bases by a reduction to guarded programs, and show that satisfiability checking of g-hybrid knowledge bases is 2-EXPTIME-complete. The DL $\mathscr{DLRO}^{-\{\leqslant\}}$ is close to $\mathscr{SHOIN}$, the Description Logic underlying OWL DL (Horrocks and Patel-Schneider, 2004a). Compared with $\mathscr{SHOIN}$, $\mathscr{DLRO}^{-\{\leqslant\}}$ does not include transitive roles and number restrictions, but does include n-ary roles and complex role expressions.

To see why a combination of rules and ontologies, as proposed in g-hybrid knowledge bases, is useful, and why the safeness conditions considered so far in the literature are not appropriate in all scenarios, consider the Description Logic ontology

$$FraternityMember \sqsubseteq Drinker \sqcap \exists hasDrinkingBuddy \cdot FraternityMember$$

which says that fraternity members are drinkers who have drinking buddies, which are also fraternity members. Now consider the logic program

$$\begin{array}{lcl} problemDrinker(X) & \leftarrow & Drinker(X), not\ socialDrinker(X) \\ socialDrinker(X) & \leftarrow & Drinker(X), not\ problemDrinker(Y), \\ & & hasDrinkingBuddy(X,Y) \\ FraternityMember(John) & \leftarrow & \end{array}$$

which says that drinkers are by default problem drinkers, unless it is known that they are social drinkers; drinkers with drinking buddies who are not problem drinkers are social drinkers; and John is a fraternity member. From the combination of the ontology and the logic program, one would expect to derive that John is a social drinker, and not a problem drinker. This logic program cannot be expressed using r-hybrid knowledge bases, or $\mathscr{DL}+log$, because the rules in the program are not weakly safe . However, the logic program is *guarded*, and thus part of a valid g-hybrid knowledge base, which has the expected consequences.

The remainder of the paper starts with an introduction to open answer set programming and Description Logics in Section 2. Section 3 defines g-hybrid knowledge bases, translates them to guarded programs when the DL $\mathscr{DLRO}^{-\{\leqslant\}}$ is considered, and provides a complexity characterization for satisfiability checking of these particular g-hybrid knowledge bases. In Section 5, we discuss the relation of g-hybrid knowledge bases with $\mathscr{DL}+log$ and other related work. We conclude and give directions for further research in Section 6.

2 Preliminaries

In this section we introduce Open Answer Set Programming, guarded programs, and the Description Logic $\mathscr{DLRO}^{-\{\leqslant\}}$.

2.1 Decidable Open Answer Set Programming

We introduce the open answer set semantics from Heymans *et al.* (2005a, 2006c), modified as in Heymans *et al.* (2006a) such that it does not assume uniqueness of names by default. *Constants*, *variables*, *terms*, and *atoms* are defined as usual. A *literal* is an atom $p(\vec{t})$ or a *naf-literal* $not\, p(\vec{t})$, with $\vec{t}$ a tuple of terms.[3] 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 the (in)equality predicates $=$ and $\neq$, usually written in infix notation; $t = s$ is an atom and $t \neq s$ is short for $not\, t = s$. A *regular* atom is an atom without equality. 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 α and β are finite sets of literals, $|\alpha^+| \leqslant 1$ (but α^- may be of arbitrary size), and every atom in α^+ is regular, i.e., α contains at most one positive atom, which may not contain the equality predicate.[4] The set α is the *head* of the rule and represents a disjunction of literals, whereas β is the *body* and represents a conjunction of literals. If $\alpha = \emptyset$, the rule is called a *constraint*. *Free rules* are rules of the form $q(\vec{X}) \vee not\, q(\vec{X}) \leftarrow$; they enable a choice for the inclusion of atoms in a model. We call a predicate p *free* if there is a free rule $p(\vec{X}) \vee not\, p(\vec{X}) \leftarrow$. Atoms, literals, rules, and programs that do not contain variables are *ground*.

For a literal, rule, or program o, let $cts(o), vars(o), preds(o)$ be the constants, variables, and predicates, respectively, in o. A *pre-interpretation* U for a program P is a pair (D, σ), where D is a nonempty *domain* and $\sigma : cts(P) \to D$ is a function which maps all constants in P to elements from D.[5] P_U is the ground program obtained from P by substituting every variable in P with every possible element from D and every constant c with $\sigma(c)$. For example, 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

$$\begin{array}{lcl} p(x) & \leftarrow & f(x, x) \\ p(y) & \leftarrow & f(y, x)\cdot \end{array}$$

Let $\mathcal{B}_P$ be the set of regular atoms obtained from the language of the ground program P. An *interpretation* I of a ground program P is a 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 $t = s$, we write $I \models t = s$ if s and t are equal terms. We write $I \models not\, p(\vec{t})$ if $I \not\models p(\vec{t})$, for $p(\vec{t})$ an atom. For a set of ground literals A, $I \models A$ holds 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

---

[3] We do not allow "classical" negation $\neg$; however, programs with $\neg$ can be reduced to programs without it, see e.g., Lifschitz *et al.* (2001).

[4] The condition $|\alpha^+| \leqslant 1$ makes the GL-reduct non-disjunctive, ensuring that the *immediate consequence operator* is well-defined; see Heymans *et al.* (2006c).

[5] In Heymans *et al.* (2006c), we only use the domain D which is there defined as a non-empty superset of the constants in P. This corresponds to a pre-interpretation (D, σ), where σ is the identity function on D.

model of P. For ground programs P containing *not*, the *reduct* (Inoue and Sakama, 1998) w.r.t. I is P^I, where P^I consists of $\alpha^+ \leftarrow \beta^+$ for every $\alpha \leftarrow \beta$ in P such that $I \models \mathit{not}\, \beta^-$ and $I \models \alpha^-$. I is an *answer set* of P if I is an answer set of P^I. Note that allowing negation in the head of rules leads to the loss of the *anti-chain property* (Inoue and Sakama, 1998) which states that no answer set can be a strict subset of another answer set. For example, a rule $a \vee \mathit{not}\, a \leftarrow$ has the answer sets $\emptyset$ and $\{a\}$. However, negation in the head is required to ensure 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 using an infinite preinterpretation. An *open interpretation* of a program P is a pair (U, M), where U is a preinterpretation 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 $p(\vec{x}) \in M$. 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. $P \cup \{p(\vec{X}) \vee \mathit{not}\, p(\vec{X}) \leftarrow\}$, where p is a predicate symbol not used in P and $\vec{X}$ is a tuple of variables. In the following, when we speak of satisfiability checking, we refer to satisfiability checking of predicates, unless specified otherwise.

Satisfiability checking w.r.t. the open answer set semantics is undecidable in general. In Heymans *et al.* (2006c), we identify a syntactically restricted fragment of programs, so-called *guarded programs*, for which satisfiability checking is decidable, which is shown through a reduction to guarded fixed point logic (Grädel and Walukiewicz 1999). The decidability of guarded programs relies on the presence of a *guard* in each rule, where a guard is an atom that contains all variables of the rule. Formally, a rule $r : \alpha \leftarrow \beta$ is *guarded* if there is an atom $\gamma_b \in \beta^+$ such that $\mathit{vars}(r) \subseteq \mathit{vars}(\gamma_b)$; γ_b is the *guard* of r. A program P is a *guarded program (GP)* if every nonfree rule in P is guarded. For example, a rule $a(X, Y) \leftarrow \mathit{not}\, f(X, Y)$ is not guarded, but $a(X, Y) \leftarrow g(X, Y), \mathit{not}\, f(X, Y)$ is guarded with guard $g(X, Y)$. Satisfiability checking of predicates w.r.t. guarded programs is 2-EXPTIME-complete (Heymans *et al.* 2006c) — a result that stems from the corresponding complexity in guarded fixed point logic.

2.2 *The Description Logic* $\mathscr{DLRO}^{-\{\leqslant\}}$

$\mathscr{DLR}$ (Calvanese *et al.* 1997; Baader *et al.* 2003) is a DL that supports roles of arbitrary arity, whereas most DLs only support binary roles. We introduce an extension of $\mathscr{DLR}$ with nominals, called $\mathscr{DLRO}$ (Heymans *et al.* 2006a). The basic building blocks of $\mathscr{DLRO}$ are *concept names* A and *relation names* **P**, where **P** denotes an arbitrary n-ary relation for $2 \leqslant n \leqslant n_{\max}$ and $n_{\max}$ is a given finite nonnegative integer. Role expressions **R** and concept expressions C are defined as

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

where i is between 1 and n in $(\$i/n : C)$; similarly in $\exists[\$i]\mathbf{R}$ and $\leqslant k\,[\$i]\mathbf{R}$ for $\mathbf{R}$ 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 nonempty 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):

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

Note that in $\mathcal{DLRO}$ the negation of role expressions is defined w.r.t. $\top_n^{\mathcal{I}}$ and not w.r.t. $(\Delta^{\mathcal{I}})^n$. A $\mathcal{DLRO}$ knowledge base Σ is a set of terminological axioms and role axioms, which denote subset relations between concept 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 $\mathcal{I}$ is a *model* of a knowledge base Σ (i.e. Σ is satisfied by $\mathcal{I}$) if all axioms in Σ are satisfied by $\mathcal{I}$; if Σ has a model, then Σ is *satisfiable*. A concept expression C is satisfiable w.r.t. a knowledge base Σ if there is a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$.

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

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

Therefore, in the remainder of the paper, we will restrict ourselves to nominals of the form $\{o\}$. We denote the fragment of $\mathcal{DLRO}$ without the number restriction $\leqslant k\,[\$i]\mathbf{R}$ with $\mathcal{DLRO}^{-\{\leqslant\}}$.

3 g-Hybrid knowledge bases

g-Hybrid knowledge bases are combinations of Description Logic (DL) knowledge bases and guarded logic programs (GP). They are a variant of the r-hybrid knowledge bases introduced in Rosati (2005a).

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

Note that in the above definition there are no restrictions on the use of predicate symbols. We call the atoms and literals in P that have underlying predicate symbols which correspond to concept 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, which are is the case in, e.g., the $\mathcal{DL}+log$ knowledge bases in Rosati (2006), the r-hybrid knowledge bases in Rosati (2005a), and the DL-safe rules in Motik *et al.* (2004). DL- atoms can appear in the head of rules, thereby enabling a bidirectional flow of information between the DL knowledge base and the logic program.

Example 1
Consider the $\mathcal{DLRO}^{-\{\leqslant\}}$ knowledge base Σ, 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 some drink z with a person y. Σ consists of the single axiom

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

which 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:

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

Together, Σ 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).

Given a DL interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and a ground program P, we define $\Pi(P, \mathcal{I})$ as the *projection* of P with respect to $\mathcal{I}$, which is obtained as follows: for every rule r in P,

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

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

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

Intuitively, the projection "evaluates" the program with respect to $\mathscr{I}$ by removing (evaluating) rules and DL literals consistently with $\mathscr{I}$; conceptually, this is similar to the reduct, which removes rules and negative literals consistently with an interpretation of the program.

Definition 2

Let (Σ, P) be a g-hybrid knowledge base. An interpretation of (Σ, P) is a tuple $(U, \mathscr{I}, M)$ such that

- $U = (D, \sigma)$ is a preinterpretation for P,
- $\mathscr{I} = (D, \cdot^{\mathscr{I}})$ is an interpretation of Σ,
- M is an interpretation of $\Pi(P_U, \mathscr{I})$, and
- $b^{\mathscr{I}} = \sigma(b)$ for every constant symbol b appearing both in Σ and in P.

Then, $(U = (D, \sigma), \mathscr{I}, M)$ is a *model* of a g-hybrid knowledge base (Σ, P) if $\mathscr{I}$ is a model of Σ and M is an answer set of $\Pi(P_U, \mathscr{I})$.

For p a concept expression from Σ or a predicate from P, we say that p is *satisfiable* w.r.t. (Σ, P) if there is a model $(U, \mathscr{I}, M)$ such that $p^{\mathscr{I}} \neq \emptyset$ or $p(\vec{x}) \in M$ for some $\vec{x}$ 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 σ the identity function on the constant symbols in (Σ, P). Furthermore, define $\cdot^{\mathscr{I}}$ as follows:

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

If $M = \{knowsfromAA(john, michael), problematic(john)\}$, then $(U, \mathscr{I}, M)$ is a model of this g-hybrid knowledge base. Note that the projection $\Pi(P, \mathscr{I})$ does not contain the rule $socialDrinker(john) \leftarrow .$

4 Translation to guarded logic programs

In this section, we introduce a translation of g-hybrid knowledge bases to guarded logic programs (GP) under the open answer set semantics, show that this translation preserves satisfiability, and use this translation to obtain complexity results for reasoning in g-hybrid knowledge bases. Before introducing the translation to guarded programs formally, we introduce the translation through an example.

Consider the knowledge base in Example 1. The axiom

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

translates to the constraint

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

Thus, the concept expressions on either side of the $\sqsubseteq$ symbol are associated with a new unary predicate name. For convenience, we name the new predicates according to the original concept expressions. The constraint simulates the behavior of the $\mathcal{DLRO}^{-\{\leqslant\}}$ axiom. If the left-hand side of the axiom holds and the right-hand side does not hold, there is a contradiction.

It remains to ensure that those newly introduced predicates behave according to the DL semantics. First, all the concept and role names occurring in the axiom above need to be defined as free predicates, to simulate the first-order semantics of concept and role names in DLs. In DLs, a tuple is either true or false in a given interpretation (cf., the law of the excluded middle); this behavior can be captured exactly by the free predicates:

$$\begin{aligned} socialDrinker(X) \vee not\, socialDrinker(X) &\leftarrow \\ drinks(X,Y,Z) \vee not\, drinks(X,Y,Z) &\leftarrow \end{aligned}$$

Note that concept names are translated to unary free predicates, whereas n-ary role names are translated to n-ary free predicates.

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

$$\begin{aligned} \top_1(X) \vee not\, \top_1(X) &\leftarrow \\ \top_3(X,Y,Z) \vee not\, \top_3(X,Y,Z) &\leftarrow \end{aligned}$$

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

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

To ensure that $\top_1^{\mathcal{I}} = \Delta^{\mathcal{I}}$, we add the constraint:

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

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, so that the latter rule is guarded; cf. the equivalent rule $\leftarrow not\, \top_1(X), X = X$.

We translate the nominal $\{wine\}$ to the rule

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

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 contain $\{wine\}(x)$ with $\sigma(wine) = x$ if and only if the corresponding interpretation $\{wine\}^{\mathcal{I}} = \{x\}$ for $wine^{\mathcal{I}} = x$.

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

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

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:

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

As we can see, the DL construct $\sqcap$ is translated to conjunction in the body of a rule.

The $\mathscr{DLRO}^{-\{\leqslant\}}$ role $\exists[\$1](drinks \sqcap (\$3/3 : \{wine\}))$ can be represented using the following rule:

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

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

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

We now proceed with the formal translation. The *closure* $clos(\Sigma)$ of a $\mathscr{DLRO}^{-\{\leqslant\}}$ knowledge base Σ is defined as the smallest set satisfying the following conditions:

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

We define $\Phi(\Sigma)$ as the smallest logic program satisfying the following conditions:

- For each terminological axiom $C \sqsubseteq D \in \Sigma$, $\Phi(\Sigma)$ contains 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, $\Phi(\Sigma)$ contains:
  $$\leftarrow \mathbf{R}(X_1,\dots,X_n), not\, \mathbf{S}(X_1,\dots,X_n) \tag{2}$$
- For each $\top_n \in clos(\Sigma)$, $\Phi(\Sigma)$ contains the free rule
  $$\top_n(X_1,\dots,X_n) \vee not\, \top_n(X_1,\dots,X_n) \leftarrow \tag{3}$$
  Furthermore, for each n-ary relation name $\mathbf{P} \in clos(\Sigma)$, $\Phi(\Sigma)$ contains
  $$\leftarrow \mathbf{P}(X_1,\dots,X_n), not\, \top_n(X_1,\dots,X_n) \tag{4}$$
  Intuitively, the latter rule ensures that $\mathbf{P}^{\mathscr{I}} \subseteq \top_n^{\mathscr{I}}$. In addition, $\Phi(\Sigma)$ has to contain the following constraint:
  $$\leftarrow not\, \top_1(X) \tag{5}$$
  which ensures that, for every element x in the preinterpretation, $\top_1(x)$ is true in the open answer set. The latter rule ensures that $\top_1^{\mathscr{I}} = D$ 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 each $D \in clos(\Sigma)$:

— if D is a concept nominal $\{o\}$, $\Phi(\Sigma)$ contains the following fact:

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

This fact ensures that $\{o\}(x)$ holds in any open answer set iff $x = \sigma(o) = o^{\mathscr{I}}$ for an interpretation of (Σ, P).

— if D is a concept name, $\Phi(\Sigma)$ contains

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

— if D is an n-ary relation name, $\Phi(\Sigma)$ contains

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

— if $D = \neg E$ for a concept expression E, $\Phi(\Sigma)$ contains the following rule:

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

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 $\mathbf{R}$, $\Phi(\Sigma)$ contains

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

Note that if negation would have been defined w.r.t. D^n instead of $\top_n^{\mathscr{I}}$, we would not be able to write the above as a guarded rule.

— if $D = E \sqcap F$ for concept expressions E and F, $\Phi(\Sigma)$ contains

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

— if $D = \mathbf{E} \sqcap \mathbf{F}$ for n-ary role expressions $\mathbf{E}$ and $\mathbf{F}$, $\Phi(\Sigma)$ contains

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

— if $D = (\$i/n : C)$, $\Phi(\Sigma)$ contains

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

— if $D = \exists[\$i]\mathbf{R}$, $\Phi(\Sigma)$ contains

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

The following theorem shows that this translation preserves satisfiability.

Theorem 1

Let (Σ, P) be a g-hybrid knowledge base with Σ a $\mathscr{DLRO}^{-\{\leqslant\}}$ knowledge base. Then, a predicate or concept expression p is satisfiable w.r.t. (Σ, P) iff p is satisfiable w.r.t. $\Phi(\Sigma) \cup P$.

Proof

($\Rightarrow$) Assume p is satisfiable w.r.t. (Σ, P), i.e., there exists a model $(U, \mathscr{I}, M)$ of (Σ, P), with $U = (D, \sigma)$, in which p has a nonempty extension. Now, we construct the open interpretation (V, N) of $\Phi(\Sigma, P)$ as follows: $V = (D, \sigma')$ with $\sigma' : \mathit{cts}(\Phi(\Sigma) \cup P) \to D$, and $\sigma'(x) = \sigma(x)$ for every constant symbol x from P and $\sigma'(x) = x^{\mathscr{I}}$ for every constant symbol x from Σ. Note that σ' is well defined, since, for a constant symbol x which occurs in both Σ and P, we have that $\sigma(x) = x^{\mathscr{I}}$. We define the set N as follows:

$$\begin{aligned} N = M &\cup \{C(x) \mid x \in C^{\mathscr{I}}, C \in \mathit{clos}(\Sigma)\} \\ &\cup \{\mathbf{R}(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in \mathbf{R}^{\mathscr{I}}, R \in \mathit{clos}(\Sigma)\} \end{aligned}$$

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 $\Phi(\Sigma) \cup P$ and (V, N) 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. We define the interpretation $(U, \mathscr{I}, N)$ of (Σ, P) as follows:

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

Moreover, for every constant symbol b that appears in both Σ and P, $b^{\mathscr{I}} = \sigma(b)$. As a consequence, $(U, \mathscr{I}, M)$ is an interpretation of (Σ, P).

It is easy to verify that $(U, \mathscr{I}, M)$ is a model of (Σ, P) which satisfies p. $\square$

Theorem 2

Let (Σ, P) be a g-hybrid knowledge base, where Σ is a $\mathscr{DLRO}^{-\{\leqslant\}}$ knowledge base. Then, $\Phi(\Sigma) \cup P$ is a guarded program with a size polynomial in the size of (Σ, P).

Proof

The rules in $\Phi(\Sigma)$ are obviously guarded. Since P is a guarded program, $\Phi(\Sigma) \cup P$ is a guarded program as well.

The size of $clos(\Sigma)$ is of the order $n \log n$, where n is the size of Σ. 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 $\Phi(\Sigma)$ is clearly polynomial in the size of $clos(\Sigma)$, assuming that the arity n of an added role expression is polynomial in the size of the maximal arity of role expressions in Σ. 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 Σ, the size of Σ 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. $\square$

Note that in g-hybrid knowledge bases, we consider $\mathscr{DLRO}^{-\{\leqslant\}}$, which is $\mathscr{DLRO}$ without expressions of the form $\leqslant k\,[\$i]\mathbf{R}$, since such expressions cannot be simulated with guarded programs. For example, consider the concept expression $\leqslant 1[\$1]R$, where R is a binary role. One can simulate the $\leqslant$ by negation as failure:

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

for some new q, with q defined such that there are at least two 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 guarded — there is no atom that contains X, Y_1, and Y_2. So, in general, expressing number restrictions such as $\leqslant k\,[\$i]\mathbf{R}$ is out of reach for GPs. From Theorems 1 and 2, we obtain the following corollary:

Corollary 1

Satisfiability checking w.r.t. g-hybrid knowledge bases (Σ, P), with Σ a $\mathscr{DLRO}^{-\{\leqslant\}}$ knowledge base, can be polynomially reduced to satisfiability checking w.r.t. GPs.

Since satisfiability checking w.r.t. GPs is 2-EXPTIME-complete (Heymans *et al.* 2006c), we obtain 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 guarded program. Then, a concept or role expression 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 $\mathscr{DLRO}^{-\{\leqslant\}}$ 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. □

5 Relation with $\mathscr{DL} + log$ and other related work

In Rosati (2006), so-called $\mathscr{DL} + log$ knowledge bases combine a Description Logic knowledge base with a *weakly safe* disjunctive logic program. Formally, for a particular Description Logic $\mathscr{DL}$, a $\mathscr{DL} + log$ *knowledge base* is a pair (Σ, P) where Σ is a $\mathscr{DL}$ 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 $r : \alpha \leftarrow \beta \in P$:

- $\alpha^- = \emptyset$,
- β^- does not contain DL atoms (*DL-positiveness*),
- each variable in r occurs in β^+ (*Datalog safeness*), and
- each variable in r, which occurs in a non-DL atom, occurs in a non-DL atom in β^+ (*weak safeness*).

The semantics for $\mathscr{DL} + log$ is the same as that of g-hybrid knowledge bases[6], with the following 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 to DL knowledge bases with ABoxes is straightforward.

- We do not require the *standard name assumption*, which basically says that the domain of every interpretation is essentially the same infinitely countable set 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 (Dean and Schreiber 2004), where neither the standard names assumption nor the unique names assumption applies. Note that Rosati also presented a version of hybrid knowledge bases which does not adhere to the unique name assumption in an earlier work (Rosati 2005b). However, the grounding of the program part is with the constant symbols explicitly appearing in the program or DL part only, which yields a less tight integration of the program and the DL part than in Rosati (2006) or in g-hybrid knowledge bases.
- We define an interpretation as a triple $(U, \mathscr{I}, M)$ instead of a pair $(U, \mathscr{I}')$ where $\mathscr{I}' = \mathscr{I} \cup M$; this is, however, equivalent to $\mathscr{DL} + log$.

The key differences of the two approaches are

- The programs considered in $\mathscr{DL} + log$ may have multiple positive literals in the head, whereas we allow at most one. However, we allow negative literals in the head, whereas this is not allowed in $\mathscr{DL} + log$. In addition, since DL atoms are interpreted classically, we may simulate positive DL atoms in the head through negative DL atoms in the body.
- Instead of Datalog safeness, we require *guardedness*. Whereas 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, with the exception of free rules. for example, $a(X) \leftarrow b(X), c(Y)$ is Datalog safe since X appears in $b(X)$ and Y appears in $c(Y)$, but this rule is not guarded since there is no atom that contains both X and Y. 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

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

This allows us 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 model in Example 1, *john* is drinking wine with some anonymous x and knows *michael* from the AA. More correct would be the rule

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

where we explicitly say that X and Y in the *drinks* and *knowsFromAA* relations 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 is guarded nonetheless. Thus, the resulting knowledge base is not a $\mathscr{DL}+log$ 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 $\mathscr{DL}+log$ knowledge bases as well, since $not\, A(\vec{X})$ in the body of the rule has the same effect as $A(\vec{X})$ in the head, where the latter is allowed in Rosati (2006). 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). For example, 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 to $p(X) \leftarrow \beta, not\, A(X)$, 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, and DL atoms occurring negatively in the body are equivalent to DL atoms occurring positively in the head which allows us to partly overcome our limitation of rule heads to one positive atom.
- We do not take into account ABoxes in the DL knowledge base. 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 $\mathscr{DL}+log$ knowledge bases is guaranteed if decidability of the conjunctive query containment problem is guaranteed for the DL at hand. In contrast, we relied on a translation of DLs to guarded programs for establishing decidability, and, as explained in the previous section, not all DLs (e.g., those with number restrictions) can be translated to such a GP.

We briefly mention $\mathscr{AL}$-log (Donini *et al.*, 1998), which is a predecessor of $\mathscr{DL}+log$. $\mathscr{AL}$-log considers $\mathscr{ALC}$ knowledge bases for the DL part and a set of positive Horn clauses for the program part. Every variable must appear in a positive atom in the body, and concept names are the only DL predicates, which may be used in the rules, and they may only be used in rule bodies.

---

[7] Rosati (2006) considers checking satisfiability of knowledge bases rather than satisfiability of predicates. However, the former can easily be reduced to the latter.

Hustadt *et al.* (2004) and Swift (2004) simulate reasoning in DLs with an LP formalism by using an intermediate translation to first-order clauses. In Hustadt *et al.* (2004), $\mathscr{SHIQ}$ knowledge bases are reduced to first-order formulas, to which the basic superposition calculus is applied. Swift (2004) translates $\mathscr{ALCQI}$ 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 $\mathscr{ALCQI}$. In the presence of terminological axioms, this is no longer possible since the finite model property is not guaranteed to hold.

In Levy and Rousset (1996), the DL $\mathscr{ALCNR}$ ($\mathscr{R}$ stands for role intersection) is extended with Horn clauses $q(\vec{Y}) \leftarrow p_1(\vec{X}_1), \ldots, p_n(\vec{X}_n)$, where the variables in $\vec{Y}$ must appear in $\vec{X}_1 \cup \ldots \cup \vec{X}_n$; $p_1, \ldots, p_n$ are either concept or 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. The semantics is defined through 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 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[8] of Motik *et al.* (2004), where also variables in concept atoms $A(X)$ need to appear in non-DL atoms. On the other hand, Motik *et al.* (2004) allow for the more expressive DL $\mathscr{SHOIN}(\mathbf{D})$, and the head predicates may be DL atoms as well. Finally, SWRL (Horrocks and Patel-Schneider 2004b) can be seen as an extension of Motik *et al.* (2004) without any safeness restriction, which results in the loss of decidability of the formalism. Compared to our work, we consider a slightly less expressive Description Logic, but we consider logic programs with nonmonotonic negation, and require guardedness, rather than role- or DL-safeness, to guarantee decidability.

In Eiter *et al.* (2004) *Description Logic programs* are introduced; atoms in the program component may be *dl-atoms* with which one can query the knowledge in the DL component. Such *dl-atoms* may specify information from the logic program which needs to be taken into account when evaluating the query, yielding a bi-directional flow of information. This leads to a minimal interface between the DL knowledge base and the logic program, enabling a very loose integration, based on an entailment relation. In contrast, we propose a much tighter integration between the rules and the ontology, with interaction based on single models rather than entailment. For a detailed discussion of these two kinds of interaction, we refer to de Bruijn *et al.* (2006a, 2006b).

Two recent approaches (Motik and Rosati 2007; de Bruijn *et al.* 2007) use an embedding in a nonmonotonic modal logic for integrating nonmonotonic logic programs and ontologies based on classical logic (e.g., DL). Motik and Rosati (2007) use the nonmonotonic logic of minimal knowledge and negation as failure (MKNF)

---

[8] DL-safeness is a restriction of the earlier mentioned weak safeness.

for the combination and show decidability of reasoning in case reasoning in the considered description logic is decidable, and the DL safeness condition (Motik *et al.* 2004) holds for the rules in the logic program. In our approach, we do not require such a safeness condition, but require the rules to be *guarded*, and make a semantic distinction between DL predicates and rule predicates. de Bruijn *et al.* (2007) introduce several embeddings of nonground logic programs in first-order autoepistemic logic (FO-AEL), and compare them under combination with classical theories (ontologies). However, de Bruijn *et al.* (2007) do not address the issue of decidability or reasoning of such combinations.

Finally, de Bruijn *et al.* (2006a, 2006b) use Quantified Equilibrium Logic as a single unifying language to capture different approaches to hybrid knowledge bases, including the approach presented in this paper. Although we have presented a translation of g-hybrid knowledge bases to guarded logic programs, our direct semantics is still based on two modules, relying on separate interpretations for the DL knowledge base and the logic program, whereas de Bruijn *et al.* (2006a, 2006b) define equilibrium models, which serve to give a unifying semantics to the hybrid knowledge base. The approach of de Bruijn *et al.* (2006a, 2006b) may be used to define a notion of equivalence between and may lead to new algorithms for reasoning with, g-hybrid knowledge bases.

6 Conclusions and directions for further research

We defined g-hybrid knowledge bases which combine Description Logic (DL) knowledge bases with guarded logic programs. In particular, we combined knowledge bases of the DL $\mathscr{DLRO}^{-\{\leqslant\}}$, 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 (Heymans *et al.* 2005a, 2006c). We discussed the relation with $\mathscr{DL}+log$ knowledge bases: g-hybrid knowledge bases overcome some of the limitations of $\mathscr{DL}+log$, such as the unique names assumption, Datalog safeness, and weak DL-safeness, but introduce 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* (Heymans *et al.* 2006b). This would allow for the definition of a hybrid knowledge base (Σ, P), where Σ is a $\mathscr{SHIQ}$ knowledge base and P is a conceptual logic program since $\mathscr{SHIQ}$ knowledge bases can be translated to conceptual logic programs.

Although there are known complexity bounds for several fragments of open answer set programming (OASP), including the guarded fragment considered in this paper, there are no known effective algorithms for OASP. In addition, at present, there are no implemented systems for open answer set programming. These are part of future work.

References

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

Calvanese, D., De Giacomo, G. and Lenzerini, M. 1997. Conjunctive query containment in description logics with n-ary relations. In *Proc. of the 1997 Description Logic Workshop (DL'97)*. 5–9.

de Bruijn, J., Eiter, T., Polleres, A. and Tompits, H. 2006a. On representational issues about combinations of classical theories with nonmonotonic rules. In *Proceedings of the First International Conference on Knowledge Science, Engineering and Management (KSEM'06)*. Lecture Notes in Artificial Intelligence, vol. 4092. Guilin, China. Springer-Verlag, Berlin.

de Bruijn, J., Eiter, T., Polleres, A. and Tompits, H. 2007. Embedding non-ground logic programs into autoepistemic logic for knowledge-base combination. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*. Hyderabad, India. AAAI Press, Menlo Park, CA.

de Bruijn, J., Pearce, D., Polleres, A. and Valverde, A. 2006b. A logic for hybrid rules. In *Proceedings of the Ontology and Rule Integration Workshop at the 2nd International Conference on Rules and Rule Markup Languages for the Semantic Web*, Athens, GA.

Dean, M. and Schreiber, G. (Eds). 2004. *OWL Web Ontology Language Reference*. W3C Recommendation 10 February 2004.

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

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

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP'88*. MIT Press, Cambridge, MA, 1070–1080.

Grädel, E. and Walukiewicz, I. 1999. Guarded fixed point logic. In *Proc. of LICS '99*. IEEE Computer Society, Los Alamitos, CA, 45–54.

Heymans, S., Van Nieuwenborgh, D., Fensel, D. and Vermeir, D. 2006a. Reasoning with the Description Logic $\mathscr{DLRO}^{-\{\leqslant\}}$ using Bound Guarded Programs. In *Proc. of Reasoning on the Web Workshop (RoW 2006)*, Edinburgh, UK.

Heymans, S., Van Nieuwenborgh, D. and Vermeir, D. 2005a. Guarded Open Answer Set Programming. In *LPNMR 2005*. LNAI, vol. 3662. Springer-Verlag, Berlin, 92–104.

Heymans, S., Van Nieuwenborgh, D. and Vermeir, D. 2005b. Nonmonotonic ontological and rule-based reasoning with extended conceptual logic programs. In *Proc. of ESWC 2005*. LNCS, vol. 3532. Springer-Verlag Berlin, 392–407.

Heymans, S., Van Nieuwenborgh, D. and Vermeir, D. 2006b. Conceptual logic programs. *Annals of Mathematics and Artificial Intelligence*. Special Issue on Answer Set Programming *47,* 1/2, 103–137.

Heymans, S., Van Nieuwenborgh, D. and Vermeir, D. 2006c. Open answer set programming with guarded programs. *ACM Transactions on Computational Logic (TOCL)*. In press.

Horrocks, I. and Patel-Schneider, P. 2004a. Reducing OWL entailment to description logic satisfiability. *Journal of Web Semantics 1,* 4, 345–357.

Horrocks, I. and Patel-Schneider, P. F. 2004b. A proposal for an OWL rules language. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*. ACM, New York, 723–731.

Hustadt, U., Motik, B. and Sattler, U. 2004. Reducing SHIQ^{-} description logic to disjunctive logic programs. In *Proceedings of the 9th International Conference on Knowledge Representation and Reasoning KR2004*, Whistler, Canada, 152–162.

Inoue, K. and Sakama, C. 1998. Negation as failure in the head. *Journal of Logic Programming 35,* 1, 39–78.

Levy, A. Y. and Rousset, M. 1996. CARIN: A representation language combining horn rules and description logics. In *Proc. of ECAI'96*. 323–327.

Lifschitz, V., Pearce, D. and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic 2,* 4, 526–541.

Motik, B. and Rosati, R. 2007. A faithful integration of description logics with logic programming. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, Hyderabad, India.

Motik, B., Sattler, U. and Studer, R. 2004. Query answering for OWL-DL with rules. In *Proc. of ISWC 2004*. LNCS, vol. 3298. Springer-Verlag, Berlin, 549–563.

Rosati, R. 2005a. On the decidability and complexity of integrating ontologies and rules. *Web Semantics 3,* 1, 41–60.

Rosati, R. 2005b. 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)*. LNCS, vol. 3703. Springer-Verlag, Berlin, 50–64.

Rosati, R. 2006. 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)*. AAAI Press, Menlo Park, CA, 68–78.

Swift, T. 2004. Deduction in ontologies via answer set programming. In *LPNMR*, V. Lifschitz and I. Niemelä, Eds. LNCS, vol. 2923. Springer-Verlag, Berlin, 275–288.