Deciding the Loosely Guarded Fragment and Querying Its Horn Fragment Using Resolution

Sen Zheng, Renate A. Schmidt
Department of Computer Science, University of Manchester
Oxford Road, Manchester M13 9PL, UK

Abstract

We consider the following query answering problem: Given a Boolean conjunctive query and a theory in the Horn loosely guarded fragment, the aim is to determine whether the query is entailed by the theory. In this paper, we present a resolution decision procedure for the loosely guarded fragment, and use such a procedure to answer Boolean conjunctive queries against the Horn loosely guarded fragment. The Horn loosely guarded fragment subsumes classes of rules that are prevalent in ontology-based query answering, such as Horn $\mathcal{ALCHOI}$ and guarded existential rules. Additionally, we identify star queries and cloud queries, which using our procedure, can be answered against the loosely guarded fragment.

Introduction

Our motivation of considering query answering problem stems from ontology-based data access (OBDA) systems (Xiao et al. 2018), which have attracted much recent attention in the knowledge representation and database communities. In particular, since the Horn loosely guarded fragment subsumes mainstream rules in OBDA systems, such as Horn $\mathcal{ALCHOI}$ (Baader et al. 2017) and the guarded existential rules (a.k.a. guarded TGDs) (Calì, Gottlob, and Lukasiewicz 2012), our interest is the development of a practical procedure for answering queries against the Horn loosely guarded fragment (van Benthem 1997).

To retrieve information from OBDA systems, the main querying mechanisms are (Boolean) conjunctive queries. Given a Boolean conjunctive query (BCQ) q, a set Σ of rules and a database $\mathcal{D}$, checking whether $\Sigma \cup \mathcal{D} \models q$ is equivalent to checking whether $\Sigma \cup \mathcal{D} \cup \neg q \models \bot$, so that the problem of answering BCQ can be reduced to deciding satisfiability. Such BCQ answering problems can be recast as query containment/entailment/evaluation problems in database research (Baget et al. 2011), constraint satisfaction problems and homomorphism mapping problems in general AI research (Vardi 2000). Although finding answers for queries is also an important problem, Boolean conjunctive query answering is widely studied (Baget et al. 2011; Bárány, Gottlob, and Otto 2010; Calì, Gottlob, and Kifer 2013; Calì, Gottlob, and Lukasiewicz 2012; Glimm et al. 2008; Gottlob, Pieris, and Tendera 2013). In this paper, we particularly focus on an open problem, namely BCQ answering for the Horn loosely guarded fragment.

The complexity of BCQ answering for the guarded fragment is 2EXPTIME-complete (Bárány, Gottlob, and Otto 2010), and satisfiability checking for the clique-guarded negation fragment, which subsumes both BCQs and the loosely guarded fragment, is also 2EXPTIME-complete (Bárány, ten Cate, and Segoufin 2015). These complexity results show that BCQ answering for the Horn (loosely) guarded fragment is decidable, however, as yet there is no practical (i.e., implementable) procedure.

Let us give a quick review of our settings and explain why we start our investigation for query answering with deciding the loosely guarded fragment (LGF). A *loosely guarded quantified formula* (van Benthem 1997; Grädel 1999) has the form $\forall \overline{x}(G_1 \wedge \ldots \wedge G_n \rightarrow F)$ where $G_1, \ldots, G_n$ are atoms that are called *guards*, F is a loosely guarded formula where i) all free variables of F occur in $G_1 \wedge \ldots \wedge G_n$, and ii) the variables in $G_1, \ldots, G_n$ are either free, or they co-occur with each other in a G_i among the $G_1, \ldots, G_n$. The Horn fragment of LGF will be referred to Horn LGF. A *Boolean conjunctive query* is a first-order formula of the form $q = \exists \overline{x} \varphi(\overline{x})$ where φ is a conjunction of atoms containing only constants and variables. One can obtain a *query clause* Q by simply negating a BCQ. Hence, Q is a negative clause containing no compound terms.

To answer BCQs over Horn LGF, we begin with deciding LGF, since it can be observed that i) some loosely guarded formulae are naturally cyclic BCQs (Bernstein and Chiu 1981) because a loosely guarded formula F allows multiple guards, and all variables in the clausal form of F (loosely guarded clauses) co-occur with each other in one of the guards. E.g., the loosely guarded formula $F = \exists xyz(A_1xy \wedge A_2yz \wedge A_3xz)$ is a cyclic BCQ. In fact, it shows that the conjunctive queries with the hyper-tree width property are strongly connected to LGF (Gottlob, Leone, and Scarcello 2003); ii) a loosely guarded formula allow variables chaining multiple literals (*chained variables*), as in F, which can cause term depth increase during reasoning (see Example 1). We see that understanding the handling of chained variables in LGF helps us handle chained variables in query clauses. Hence, in this paper, we first provide a decision procedure to decide LGF. Subsequently we show that such a procedure can be extended to answer BCQs over Horn LGF and answer restricted BCQs over LGF.

Considering the only existing decision procedure for answering BCQs over (Horn) LGF essentially aims at the theoretical analysis (Bárány, ten Cate, and Segoufin 2015), our focus is on devising a practical decision procedure, so that one can implement our procedure as query engines. We build our procedure using resolution in the framework of (Bachmair and Ganzinger 2001), which is a standard in the area of automated reasoning and provides basis for most first-order logic reasoners such as Spass (Weidenbach et al. 2009), Vampire (Riazanov and Voronkov 2001) and E (Schulz 2013). References for using resolution with refinement as practical decision procedures are (Ganzinger et al. 1998; Ganzinger and de Nivelle 1999; Hustadt and Schmidt 1999; Hustadt 1999; Hustadt and Schmidt 1997; Bachmair, Ganzinger, and Waldmann 1993).

In resolution-based reasoning, the main challenges to avoid non-termination are: i) avoiding unlimited growth of the number of literals in resolvents, and ii) avoiding unlimited growth of term depth in resolvents. The former can be tackled by using the property of loosely guarded clauses C: Guards in C contain all variables of C, so that using our inference system, the number of literals cannot increase indefinitely in the resolvents. As for the latter, consider this example:

Example 1. *Q is a query clause and C_1, C_2 are loosely guarded clauses:*

$$Q = \neg A_1(x, y) \vee \neg A_2(y, z),$$
$$C_1 = A_1(f(x_1, y_1), x_1) \vee B_1(g(x_1, y_1)) \vee \neg G_1(x_1, y_1),$$
$$C_2 = A_2(h(x_2, y_2), x_2) \vee \neg G_2(x_2, y_2)$$

Performing resolution among Q, C_1, C_2 derives $R = B_1(g(h(x_2, y_2)), y_1) \vee \neg G_1(h(x_2, y_2), y_1) \vee \neg G_2(x_2, y_2)$. In R, $g(h(x_2, y_2))$ is deeper than all the terms in Q, C_1 and C_2. This happens when a query clause Q contains a chained variable y where: i) y needs to be unified with a variable x_1 in C_1 and a non-ground compound term $h(x_2, y_2)$ in C_2 at the same time and, ii) x_1 occurs in a non-ground compound term $g(x_1, y_1)$ in C_1. However, such term depth increase can be avoided if we perform resolution on Q and C_1 first. We introduce the top variable technique in its corresponding section to show how the term depth increase in R can be prevented.

Our resolution decision procedure for LGF is a variation of procedures presented in (de Nivelle and de Rijke 2003) and (Ganzinger and de Nivelle 1999). Like (Ganzinger and de Nivelle 1999), but unlike (de Nivelle and de Rijke 2003), which uses a non-liftable ordering, our procedure uses admissible and liftable orderings with selection, and is consistent with the resolution framework of (Bachmair and Ganzinger 2001). Inspired by the 'MAXVAR' technique in (de Nivelle and de Rijke 2003) and the partial hyper-resolution technique from (Ganzinger and de Nivelle 1999), we use the top variable technique to avoid term depth increase. (Ganzinger and de Nivelle 1999) mainly focuses on deciding the guarded fragment, and refers to a manuscript version of (de Nivelle and de Rijke 2003) for technical details about using the 'MAXVAR' technique to decide LGF. As for 'MAXVAR', unlike (de Nivelle and de Rijke 2003), we use a unification-first approach to identify top variables while (de Nivelle and de Rijke 2003) finds 'MAXVAR' by variable depth first, then applies a specific unification algorithm. It turns out that our approach allows top variables ('MAXVAR') being easily identified since no specific permutation and unification algorithms are needed. Further, we embed the top variable technique into the framework of (Bachmair and Ganzinger 2001) as a selection function (top selection) with completeness proofs, so that, together by using liftable orderings, one immediately benefits from being able to use the notions of redundancy in that framework. Additionally, we generalise the pre-conditions of 'MAXVAR', so that the top technique can be applied to a larger class than LGF (any clauses that satisfy conditions of query pair clauses), including queries, so that one can use the top technique to query other fragments of first-order logic to avoid variable depth growth as well. This makes our procedure applicable for the problems of BCQ answering for Horn LGF, and the star/cloud query answering for LGF.

Without hurting the result of this paper, we discuss variable depth, rather than term depth for the termination result. This holds because a term depth can grow infinitely if and only if a variable depth grows infinitely.

The contributions of this paper are:

- A variation of the resolution-based decision procedure for LGF in (de Nivelle and de Rijke 2003), situated in the framework of (Bachmair and Ganzinger 2001).
- By expanding the top variable technique to query pair clauses, this procedure provides the basis for a practical decision procedure for answering BCQs over Horn LGF.
- We identify star queries and cloud queries so that one can use our procedure to answer these queries against LGF.

Preliminaries

Let **C**, **F**, **P** denote pairwise disjoint sets of *constant symbols*, *function symbols* and *predicate symbols*, respectively. The definition of *(compound/ground) term*, *atom*, *literal*, *clause*, *expression*, *argument*, *unifier*, *most general unifier* (mgu) and *simultaneous mgu* are defined as usual in automated reasoning (see e.g., (Bachmair and Ganzinger 2001) for details). A literal L is a *non-ground compound literal* if L contains at least one non-ground compound term. Let $\overline{x}, \overline{A}, \mathcal{C}$ denote a sequence of variables, a sequence of atoms and a set of clauses, respectively. Let $var(t)$, $var(C)$ and $var(\overline{A_n})$ denote sets of variables in a term t, a clause C and a sequence of atoms $\overline{A_n}$, respectively.

The variable depth of a term t, denoted as $vdp(t)$, is defined as follows: i) if t is ground, then $vdp(t) = -1$, and if t is not ground, then ii) if t is a variable, then $vdp(t) = 0$, and iii) if t is a non-ground compound term $f(u_1, \ldots, u_n)$, then $vdp(t) = 1 + max(\{vdp(u_i) \mid 1 \leq i \leq n\})$. A term t is *flat* if

$vdp(t) \leq 0$. A term t is *simple* if $vdp(t) \leq 1$. A *flat* (*simple*) *atom, literal* and *clause* is an atom, a literal and a clause such that every term in it is flat (simple). We say a term t is a simple non-ground compound term if $vdp(t) = 1$. Assume $C = \neg A_1 \vee \ldots \vee \neg A_n \vee D$ is a clause where $\neg A_1 \vee \ldots \vee \neg A_n$ are flat literal. Then $x \in var(A_1, \ldots, A_n)$ can be: i) a *chained variable*: x occurs in two literals A_i, A_j among $A_1, \ldots, A_n$ such that $var(A_i) \not\subseteq var(A_j)$, $var(A_j) \not\subseteq var(A_i)$, and $x \in var(A_i) \cap var(A_j)$, and ii) an *isolated variable*: x is not chained. In Q of Example 1, y is a chained variable and x, z are isolated variables. By the *length* of a clause, we mean the number of literals that occur in a clause, and by the *depth* of a clause, we mean the deepest variable depth of a clause. In this paper, we assume the input clauses (formulae) are of fixed-length and fixed-width.

A *weakly covering* term is a compound term t such that for every non-ground, compound subterm s of t, it is the case that $var(s) = var(t)$ (Fermüller et al. 1993). A literal L is weakly covering if each argument of L is either a ground term, a variable, or a weakly covering term t, such that $var(t) = var(L)$. A clause C is weakly covering if each term t in C is either a ground term, a variable, or a weakly covering term such that $var(t) = var(C)$. E.g., the clause $C_1 = \neg A_1(fxyza, x, y, ga) \vee A_2xyz$ is a weakly covering clause since the only non-ground compound term $fxyza$ satisfies that $var(fxyza) = var(C_1)$, however, the clause $C_2 = \neg A_1(gy, y, ga) \vee A_2(hxy)$ is not weakly covering since $var(gy) \neq var(C_2)$. Here the notion of weakly covering literals in (Fermüller et al. 1993) is extended to weakly covering clauses; reasons are given in the loosely guarded clauses section.

Recall, the *rule set* Σ denotes a set of first-order formulae and the *database* $\mathcal{D}$ denotes a set of ground atoms. A *Boolean conjunctive query* is a first-order formula of the form $q = \exists \overline{x} \varphi(\overline{x})$ where φ is a conjunction of atoms containing only constants and variables. We use the symbol Q to denote the *query clause* $\neg q$, so that we can answer BCQ satisfiability of $\Sigma \cup D \models q$ by checking whether $\Sigma \cup D \cup Q \models \bot$.

The Loosely Guarded Fragment

Definition 1. The loosely guarded fragment (LGF) *is a fragment of first-order logic without equality and function symbols, defined inductively as follows:*

1. *$\top$ and $\bot$ are in LGF.*
2. *If A is an atom, then A is in LGF.*
3. *LGF is closed under Boolean combinations.*
4. *If $F \in LGF$ and $G_1, \ldots, G_n$ are atoms, then a formula $\forall \overline{x}(G_1 \wedge \ldots \wedge G_n \rightarrow F)$ belongs to LGF if i) all free variables of F belong to $var(G_1, \ldots, G_n)$, and ii) for each variable $x \in \overline{x}$ and each variable $y \in var(G_1, \ldots, G_n)$ where $x \neq y$, x and y co-occur in a G_i. The negative literals $\neg G_1, \ldots, \neg G_n$ are called the* guards *of this formula.*

The first-order logic translation of a temporal logic formula P **until** Q is a loosely guarded formula: $\exists y(Rxy \wedge Qy \wedge \forall z((Rxz \wedge Rzy) \rightarrow Pz)))$, but the transitivity formula $\forall xyz((Rxy \wedge Ryz) \rightarrow Rxz)$ is not a loosely guarded formula since x and z do not co-occur in a guard.

We use the loosely guarded formula F in Example 2 to illustrate the clausal normal form transformation for LGF.

Example 2. $\exists y(Rxy \wedge Qy \wedge \forall z((Rxz \wedge Rzy) \rightarrow \exists x Pxy))$

The Resolution Calculus

In this section, we introduce the resolution calculus, which gives us the main termination result of this paper. The inference steps are restricted by an admissible ordering and a selection function, so that the search space can be reduced when a reasoner computes inferences. For more technical details about rules used in this paper, we refer readers to (Bachmair and Ganzinger 2001).

Let $\succ$ be a strict ordering, called a *precedence*, on the symbols in the **C**, **F** and **P**. An ordering $\succ$ is *liftable* if for all expressions E_1 and E_2 and all substitutions σ, $E_1 \succ E_2$ implies $E_1\sigma \succ E_2\sigma$. An ordering $\succ$ on literals is *admissible*, if i) it is well-founded and total on ground literals, and *liftable*, ii) $\neg A \succ A$ for all ground atoms A, and iii) if $B \succ A$, then $B \succ \neg A$ for all ground atoms A and B. A literal L is *$\succ$-maximal with respect to a clause C* if for any L' in C, $L' \not\succ L$, and L is *strictly $\succ$-maximal with respect to a clause C* if for any L' in C, $L' \not\succeq L$. A *selection function $\mathcal{S}$* selects a possibly empty set of occurrences of negative literals in a clause C with no restriction imposed. Inferences are only performed on eligible literals. A literal L is *eligible* in a clause C if either nothing is selected in the selection function $\mathcal{S}$ and L is a $\succ$-maximal literal with respect to C, or L is selected by $\mathcal{S}$.

Inferences are computed using the following rules:

Deduction: N derives N, C if C is either a resolvent or a factor of clauses in the set N.

Factors and resolvents are derived using:

Ordered factoring: $C \vee A_1 \vee A_2$ derives $(C \vee A_1)\sigma$, where i) σ is the mgu of A_1 and A_2, and ii) no literal is selected in C, and iii) $A_1\sigma$ is $\succ$-maximal with respect to $C\sigma$.

Ordered resolution with selection: $\neg A_1 \vee \ldots \vee \neg A_n \vee D$, $B_1 \vee D_1, \ldots, B_n \vee D_n$ derive $(D_1 \vee \ldots \vee D_n \vee D)\sigma$ where i) either $\neg A_1 \vee \ldots \vee \neg A_n$ are selected in D, or $n = 1$, no literal is selected, and $\neg A_1\sigma$ is $\succ$-maximal with respect to $D\sigma$, and ii) no literal is selected in $D_1, \ldots, D_n$ and $B_1\sigma, \ldots, B_n\sigma$ are strictly $\succ$-maximal with respect to $D_1\sigma, \ldots, D_n\sigma$, respectively, and iii) σ is a simultaneous mgu such that $A_1\sigma = B_1\sigma, \ldots, A_n\sigma = B_n\sigma$, and iv) $\neg A_1 \vee \ldots \vee \neg A_n \vee D, B_1 \vee D_1, \ldots, B_n \vee D_n$ are pairwise variable-disjoint.

In ordered factoring and ordered resolution, maximality is computed using *a-posteriori application* of the mgu σ. This means the maximal literal is determined after application of σ, derived from the unification algorithm applied to the premises of a rule. If the maximal literal is determined before the application of σ, we call this *a-priori application*.

Redundancy is eliminated using:

Deletion: N, C derives N if C is a tautology, or N contains a variant of C, or N contains a condensed form of C.

The 'Deletion' rule is the only rule used to eliminate redundancy, and turns out to be sufficient for the termination result. Since we employ an admissible ordering with a selection function as resolution refinement in accordance with the framework of (Bachmair and Ganzinger 2001), we can also use more sophisticated simplification rules and redundancy elimination of that framework, e.g, subsumption deletion and forward/backward subsumption.

A ground clause C is *redundant with respect to* N if there are ground instances $C_1\sigma, \dots, C_n\sigma$ of clauses in N such that $C_1\sigma, \dots, C_n\sigma \models C$ and for each i, $C \succ C_i\sigma$. A non-ground clauses C is *redundant with respect to* N if every ground instance of C is redundant with respect to N. A set of clauses N is *saturated up to redundancy* (with respect to ordered resolution and selection) if any inference from non-redundant premises in N is redundant in N (Bachmair and Ganzinger 2001).

The Decision Procedure

Now we can discuss the resolution procedure for LGF.

Clausal Normal Form Translation LGF-Trans

The clausal normal form transformation we use is similar to the one in (de Nivelle and de Rijke 2003) and (Ganzinger and de Nivelle 1999), but i) free variables are assumed to be existentially quantified since the focus is on satisfiability checking, and ii) prenex normal form and outer Skolemisation (Nonnengart and Weidenbach 2001) are used. Though outer Skolemisation may introduce Skolem functions of higher arities than inner/standard Skolemisation, outer Skolemisation turns out to be critical to guarantee that output clauses have the weakly covering property.

We use *LGF-Trans* to denote the clausal normal form transformation below. Using F in Example 2, one can obtain a set of loosely guarded clauses via the following steps:

i) Add existential quantifiers to all free variables in F:

$\exists xy(Rxy \wedge Qy \wedge \forall z((Rxz \wedge Rzy) \rightarrow \exists x Pxy))$.

ii) Rewrite $\rightarrow$ and $\leftrightarrow$ using conjunction, disjunction and negation, and transform F into negation normal form, obtaining the formula F_{nnf}:

$\exists xy(Rxy \wedge Qy \wedge \forall z(\neg Rxz \vee \neg Rzy \vee \exists x Pxy))$.

iii) Apply optimised structural transformation to F_{nnf}, that introduces fresh predicate symbols (Q_1) for universally quantified subformulae ($\forall z(\neg Rxz \vee \neg Rzy \vee \exists Pxy)$), obtaining the formula F_{str}:

$\exists xy(Rxy \wedge Qy \wedge Q_1xy) \wedge$

$\forall xy(\neg Q_1xy \vee \forall z(\neg Rxz \vee \neg Rzy \vee \exists x Pxy))$.

iv) Find $\exists xy \forall uvw \exists x'((Rxy \wedge Qy \wedge Q_1xy) \wedge (\neg Q_1uv \vee \neg Ruw \vee \neg Rwv \vee Px'v))$ as the prenex normal form of F_{str}, and apply outer Skolemisation: if $\forall \overline{x}$ is the subsequence of all universal quantifiers of the φ-prefix of subformula $\exists y\varphi$ of φ, then $\varphi[y/f(\overline{x})]$ is the outer Skolemisation of $\exists y\varphi$. Skolem terms $a, b, fxyz$ are introduced, obtaining F_{sko}:

$Rab \ \wedge \ Qb \ \wedge \ Q_1ab \wedge$

$\forall xyz(\neg Q_1xy \vee \neg Rxz \vee \neg Rzy \vee P(fxyz, y))$

v) Drop all universal quantifiers and transform F_{sko} into conjunctive normal form, obtaining loosely guarded clauses:

Rab, Qb, Q_1ab, $\neg Q_1xy \vee \neg Rxz \vee \neg Rzy \vee P(fxyz, y)$

Loosely Guarded Clauses

We now describe loosely guarded clauses and their properties.

Definition 2. *A* loosely guarded clause (LGC) *C is a clause satisfying the following conditions:*

1. *C is simple and weakly covering, and*
2. *if C is non-ground, then there is a set of negative literals $\neg G_1, \dots, \neg G_n$ in C that are flat. Then $\neg G_1, \dots, \neg G_n$ are called the* guards *of C, such that each pair of variables in C co-occur in at least one of the guards.*

We can immediately see that a ground clause is an LGC.

Proposition 1. *Using* LGF-Trans*, every loosely guarded formula can be transformed into a set of LGCs.*

The class of LGCs strictly subsumes LGF since function symbols are allowed. If using an admissible ordering in which function symbols have higher precedence than other symbols, then non-ground compound terms in an LGC C are always larger than variables in C due to the weakly covering property. E.g., with a lexicographic path ordering $\succ_{lpo}$ (Dershowitz 1982), considering an LGC $C = \neg A_1xy \vee \neg A_2yz \vee \neg A_3xz \vee D(fxyz)$ and an arbitrary substitution σ, D is $\succ_{lpo}$-maximal with respect to C if $D\sigma$ is $\succ_{lpo}$-maximal with respect to $C\sigma$ since $var(D) = var(C)$. This shows that when determining the maximal literal in an LGC, the result of a-priori application follows the result of the a-posteriori application. To avoid the overhead of pre-computating the mgu using the a-posteriori application (Fermüller et al. 1993), we use the a-priori application. We show this result in Lemma 1:

Lemma 1. *Assume C is a weakly covering clause containing a non-ground compound literal L and σ is an arbitrary substitution. Using any admissible ordering $\succ$ with a precedence that function symbols are larger than other symbols, L is $\succ$-maximal with respect to C if $L\sigma$ is $\succ$-maximal with respect to $C\sigma$.*

An obvious property of a weakly covering clause C is that all non-ground terms and literals in C are also weakly covering. Formally stated as:

Lemma 2. *If a clause C is weakly covering, then for each non-ground compound term t and each non-ground compound literal L occur in C, $var(t) = var(L) = var(C)$.*

The Top Variable Technique

Before discussing the resolution calculus for LGCs, we introduce the top variable technique, as a variation of the 'MAXVAR' technique in (de Nivelle and de Rijke 2003). The top variable technique is a look-ahead approach to prevent variable depth increase in the resolvents: Suppose we have a set of clauses that can lead to variable depth increase in the resolvent. Using the top variable technique, we first identify clauses that lead to the potentially deepest terms, and then perform resolution on those clauses first. Next we perform inference on the rest of the clauses. In such a manner of performing resolution, we show that no variable depth increase occurs in the resolvents.

Definition 3 (Query Pair Clauses)**.** *Let $\overline{A_n}$, $\overline{B_n}$ be a sequence of atoms $A_1, \ldots, A_n$ and a sequence of weakly covering atoms $B_1, \ldots, B_n$, respectively. $(\overline{A_n}, \overline{B_n})$ is a* query pair *if they satisfy these conditions:*

1. *$\overline{A_n}$ is flat and non-ground, and $\overline{B_n}$ is simple.*
2. *Each $B_i \in \overline{B_n}$ either is a non-ground compound literal or is a ground literal.*
3. *$var(\overline{A_n}) \cap var(\overline{B_n}) = \emptyset$, and $B_1, \ldots, B_n$ are pairwise variable disjoint.*
4. *There exists an mgu (simultaneous mgu if $n > 1$) σ such that for each $A_i \in \overline{A_n}$, $B_i \in \overline{B_n}$, $A_i\sigma = B_i\sigma$.*

Let $(\overline{A_n}, \overline{B_n})$ be a query pair. Query pair clauses *for a query pair $(\overline{A_n}, \overline{B_n})$ is a set of clauses: $C = \neg A_1 \vee \ldots \vee \neg A_n \vee D$, $C_1 = B_1 \vee D_1$, …, $C_n = B_n \vee D_n$ where D is a flat clause and $D_1, \ldots, D_n$ are simple clauses.*

From now on, we also use mgu to denote the simultaneous mgu. To find top variables in $\overline{A_n}$ in a query pair, one needs to find the mgu between $\overline{A_n}$ and $\overline{B_n}$ to identify the variable orderings over $var(\overline{A_n})$.

Definition 4 (Variable Ordering)**.** *Let $(\overline{A_n}, \overline{B_n})$ be a query pair and let an mgu σ satisfy Condition 4 in Definition 3.*

By $>_v, =_v$ we denote a variable ordering *over $var(\overline{A_n})$, which is defined by: for $x, y \in var(\overline{A_n})$, i) $x >_v y$ iff $vdp(x\sigma) > vdp(y\sigma)$, ii) $x =_v y$ iff $vdp(x\sigma) = vdp(y\sigma)$.*

Using the notion of variable orderings, we define top variables, and show the existence of top variables in query pairs:

Definition 5 (Top variable)**.** *Given a query pair $(\overline{A_n}, \overline{B_n})$, a variable $x \in var(\overline{A_n})$ is a* top variable *iff for each $y \in var(\overline{A_n})$, $x >_v y$ or $x =_v y$.*

Proposition 2. *Let $(\overline{A_n}, \overline{B_n})$ be a query pair. Then at least one of the variables in $\overline{A_n}$ is a top variable.*

The idea behind the top variable technique is finding the potentially deepest term of query pair clauses. To realise it, we first apply the unification algorithm, then make the literals in the main premise containing the potentially deepest terms eligible literals. In Example 1, the mgu $\sigma = \{x/f(hx_2y_2, y_1), y/hx_2y_2, z/x_2, x_1/hx_2y_2\}$, thus applying resolution among Q, C_1, C_2 derives $B_1(g(hx_2y_2), y_1)$, in which the first argument is deeper than all terms in Q, C_1, C_2. Now we use top variables to find the deepest terms. First we find top variables in Q: since $vdp(x\sigma) > vdp(y\sigma) > vdp(z\sigma)$, $x >_v y >_v z$. Since x is the top variable (potentially the deepest term), we make the literal A_1 eligible since x only occurs in A_1, then applying resolution on clauses Q, C_1, deriving the resolvent $C_3 = \neg A_2(x_1, z) \vee B_1(gx_1y_1) \vee \neg G_1(x_1, y_1)$. Though C_3 is not weakly covering, there is no variable depth increase in C_3.

Let $(\overline{A_n}, \overline{B_n})$ be a query pair, and assume query pair clauses $C, C_1, \ldots, C_n$ such that $C = \neg A_1 \vee \ldots \vee \neg A_n \vee D$, as the *main premise*, $C_1 = B_1 \vee D_1$, …, $C_n = B_n \vee D_n$ as the *side premises*, where D is flat and $D_1, \ldots, D_n$ are simple. Assume $A_1, \ldots, A_t$ is a sequence of atoms containing top variables and the respective counterparts are $B_1, \ldots, B_t$, which occur in $C_1, \ldots, C_t$, respectively, and σ is the mgu such that $A_i\sigma = B_i\sigma$ where $1 \leq i \leq t \leq n$. We denote *Res* as an application of resolution among the clauses $C, C_1, \ldots, C_t$:

$$\frac{B_1 \vee D_1, \ldots, B_t \vee D_t \quad \neg A_1 \vee \ldots \vee \neg A_t \vee \ldots \vee \neg A_n \vee D}{(D_1 \vee \ldots \vee D_t \vee \neg A_{t+1} \vee \ldots \vee \neg A_n \vee D)\sigma}$$

Given two expressions $A(\ldots, t, \ldots)$ and $B(\ldots, u, \ldots)$, we say t *matches* u if the argument position of t in A is the same as the argument position of u in B. We show how top variables in $var(\overline{A_n})$ match, the result is stated as:

Lemma 3. *In an application of* Res*,*

1. *a top variable matches either a ground term or a non-ground compound term, and*
2. *a non-ground compound term matches a top variable.*

Based on the matching in Lemma 3, we now show properties of mgus in *Res*:

Lemma 4. *In an application of* Res*, these conditions hold:*

1. *The mgu assigns to top variables either simple non-ground compound terms or ground terms.*
2. *The mgu assigns to non-top variables in the main premise either variables or ground terms.*
3. *The mgu assigns to variables in the side premises either variables or ground terms.*

Using Lemma 4, we give Theorem 1, which says that, for query pair clauses, only resolving literals that contain the potentially deepest terms does not lead to variable depth growth in the resolvents.

Theorem 1. *In an application of* Res*, no variable depth growth occurs in the resolvents of a set of query pair clauses.*

Resolution Refinement LGC-Refine

Now we formally describe the orderings and selection as refinement to decide LGF. One can use any admissible ordering that satisfies the conditions in *LGC-Refine*. Here a lexicographic path ordering $\succ_{lpo}$ (Dershowitz 1982) is used.

Definition 6 (LGC-Refine)**.** *Let* LGC-Refine *denote the refinement: A lexicographic path ordering $\succ_{lpo}$ based on a precedence $f > a > p$ for $f \in \mathbf{F}$, $a \in \mathbf{C}$ and $p \in \mathbf{P}$, and a selection function such that the following conditions hold:*

1. *If a clause contains negative non-ground compound literals, then at least one of these literals is selected.*
2. *If a clause contains no negative non-ground compound literal, but there are positive non-ground compound literals, then the maximality principle with respect to $\succ_{lpo}$ is applied to determine the eligible literals.*
3. *If a clause contains no non-ground compound literals, select all the negative literals containing top variables.*

We use *top selection* to denote selection based on the top variable technique. Condition 3 in *LGC-Refine* implies that top selection is imposed not only to guards. E.g., although $\neg B(x)$ is not a guard in $\neg A(x, y) \vee \neg B(x)$, top selection would select both A and B if x is a top variable.

The Resolution Calculus LGC-Res

Now we discuss how resolution with *LGC-Refine* performs over LGCs. We use the notation *LGC-Res* to denote the calculus consisting of the following: the 'Deduction' rule, ordered factoring and ordered resolution with selection refined by *LGC-Refine*, and the 'Deletion' rule. When applying *LGC-Res*, the 'Deletion' rule and the 'Deduction' rule are used whenever they are applicable. As usual, we assume the input clauses (after condensation and modulo variable renaming) are a finite set of fixed LGCs.

First we discuss the ordered resolution rule. We use *Res*$'$ to denote the resolution rule when one of the premises satisfy Condition 3 in *LGC-Refine*.

Let $C = \neg A_1 \vee \ldots \vee \neg A_n \vee D$ be a flat LGC, as the *main premise*, and $C_i = B_i \vee D_i$ be a set of LGCs, as the *side premises*. Let $\overline{A_{t(n)}}, \overline{B_t}$ denote $A_1, \ldots, A_{t(n)}$ and $B_1, \ldots, B_t$ where $1 \leq t \leq n$, respectively. Using *LGC-Refine*, *Res*$'$ is performed as:

$$\frac{B_1 \vee D_1, \ldots, B_t \vee D_t \quad \neg A_1 \vee \ldots \vee \neg A_t \vee \ldots \vee \neg A_n \vee D}{(D_1 \vee \ldots \vee D_t \vee \neg A_{t+1} \vee \ldots \vee \neg A_n \vee D)\sigma}$$

where i) C is non-ground and D is positive, ii) each $A_i \in \overline{A_t}$ contains at least one top variable and each $B_i \in \overline{B_t}$ is strictly $\succ_{lpo}$-maximal with respect to C_i, respectively, and σ is the mgu such that $A_i\sigma = B_i\sigma$ where $1 \leq i \leq t$, iii) $C, C_1, \ldots, C_t$ are pairwise variable disjoint.

Since the premises in *Res*$'$ (LGCs) satisfy conditions of query pair clauses, we can inherit results of *Res*. The particularities in *Res*$'$ are: i) all premises are weakly covering clauses rather than only literals are weakly covering, and ii) each premise contains a set of guards.

First we show that using *Res*$'$, every resolvent is simple:

Corollary 1. *In an application of* Res$'$*, the resolvents of a set of LGCs are simple clauses.*

To show the resolvents in *Res*$'$ are LGCs, we need to discuss some unique properties in *Res*$'$ comparing to *Res*:

Lemma 5. *In an application of* Res$'$*, if we use notions from* Res$'$*, and let x be a top variable in $A_1, \ldots, A_s$ ($s \leq t$). Then*

1. *$var(A_1, \ldots, A_s) = var(C)$, and*
2. *$var(x\sigma) = var(C\sigma)$, and*
3. *$var(x\sigma) = var(y\sigma)$ if x, y are distinct top variables.*

Now we can show the resolvents in *Res*$'$ have are indeed LGCs: they are weakly covering and contain a set of guards.

Lemma 6. *In an application of* Res$'$*, the resolvents of a set of LGCs are LGCs.*

It remains to consider other possibilities in *LGC-Res*. In particular, we discuss situations that are not covered by *Res*$'$ such that there is no premise satisfying Condition 3 in *LGF-Refine*: the negative premise satisfies Condition 1 in *LGF-Refine* or is ground. This is the case when the ordered resolution with selection is naturally reduced to a binary case.

Lemma 7. *In an application of* LGC-Res*, the factors of LGCs are LGCs, and the resolvents of LGCs are LGCs.*

Now we can show the main result of this section:

Theorem 2. *Given a set of LGCs, using* LGC-Res*, all inferred clauses are LGCs.*

So far we have shown that using *LGC-Res*, the resolvents of LGCs are LGCs. Since an LGC is a simple clause, there is no variable depth increase during the inference. We still need to consider that using *LGC-Res*, the length of the resolvents cannot be infinitely long:

Lemma 8. *In an application of* LGC-Res*, the number of variables in derived clauses is no more than the number of variables of one of the premises of these derived clauses.*

Refutational Completeness and Termination

This section we give the refutational completeness and termination results of applications of *LGC-Res* over LGCs. For refutational completeness result, we particularly show that the top selection used in *LGC-Refine* is compatible within the framework of (Bachmair and Ganzinger 2001).

Theorem 3 (Refutational Completeness)**.** *Let N be a set of clauses that are saturated up to redundancy under* LGC-Res*, then N is unsatisfiable iff N contains the empty clause.*

Let *LGF-Res* denote the combination of the clausal transformation *LGF-Trans* and the resolution calculus *LGC-Res* with refinement *LGC-Refine*. Now we give the first main result of this paper:

Theorem 4. LGF-Res *decides LGF.*

Querying Horn LGF and LGF

In this section, we aim to check whether $\Sigma \cup \mathcal{D} \cup Q \models \bot$ where Σ are formulae in (Horn) LGF, $\mathcal{D}$ is a set of ground atoms and Q is a query clause. Because Σ and $\mathcal{D}$ can be transformed into (Horn) LGCs using *LGF-Trans*, the aim now is to check whether $\mathcal{C} \cup Q \models \bot$ where $\mathcal{C}$ is a set of (Horn) LGCs and Q is a query clause. In particular, we assume Q is a fixed query clause. We show that when either Q is restricted to star/cloud queries, or Σ is restricted to Horn LGF, our procedure guarantees termination.

Since a query clause Q contains no non-ground compound terms, Q satisfies Condition 3 in *LGC-Refine*, thus no particular new refinement for Q is needed. Hence we still can use *LGC-Refine* as refinement for inference rules. However, *LGC-Res* does not contain a rule to compute resolvents of a query clause and a set of LGCs. Using *LGC-Refine*, a query clause Q and a set of LGCs $C_1, \ldots, C_n$ satisfy conditions of query pair clauses, thus we apply *Res* to compute resolvents of Q and $C_1, \ldots, C_n$. We use *Query-Res* to denote *LGF-Res* and *Res* using refinement *LGC-Refine*. Since no positive factoring can be applied to query clauses, we only discuss how resolution is performed on query clauses.

According to Theorem 1, the following result holds:

Corollary 2. *In the application of* Res*, there is no variable depth growth in the resolvents of a query clause and a set of LGCs, thus the resolvents are simple clauses.*

Querying Horn LGF

It follows from Theorem 2 that using *Query-Res*, the resolvents of a set of LGCs are LGCs. We now discuss the resolvent of a query clause and a set of LGCs. Corollary 2 shows that a resolvent R of a query clause and a set of LGCs is a simple clause. However, this simple clause R can be neither an LGC nor a query clause. In Example 1, one can obtain $C_3 = \neg A_2(x_1, z) \vee B_1(gx_1y_1) \vee \neg G_1(x_1, y_1)$ using *Res* with *LGC-Refine*. Although C_3 is simple, C_3 is neither an LGC (not weakly covering), nor a query clause (not flat).

We observe that by disallowing multiple positive non-ground compound literals in LGCs, using *Res*, the resolvent R of a query clause and Horn LGCs is a negative clause containing no non-ground compound term, thus R is a query clause. If we change C_1 in Example 1 to a Horn LGC $C_1' = A_1(fx_1y_1, x_1) \vee \neg G_1(x_1, y_1)$, then by applying *Res* with refinement *LGC-Refine*, since x is a top variable, resolution between Q, C_1' derives $C_3' = \neg A_2(x_1, z) \vee \neg G_1(x_1, y_1)$, which is a query clause. According to *LGC-Refine*, further inference between C_2 and C_3' requires another positive premise that contains a positive literal G_1 that is either ground or contains non-ground compound terms. Hence, no inference is performed between C_2 and C_3'.

We show that *Query-Res* can decide $\mathcal{C} \cup Q \models \perp$ where $\mathcal{C}$ are formulae in Horn LGF and Q is a query clause. Notice that ground atoms $\mathcal{D}$ are immediately in Horn LGF. A *Horn loosely guarded clause* (Horn LGC) is an LGC that contains at most one positive literal. The *Horn loosely guarded fragment* (Horn LGF) is a subset of LGF that can be transformed into a set of Horn LGCs using *LGF-Trans*.

Using *Res*, when query pair clauses are query clauses and a set of Horn LGCs, the resolvents are query clauses:

Lemma 9. *In an application of* Res*, the resolvents of Horn LGCs and a query clause are query clauses.*

It turns out the proof of Lemma 9 does not require the premises to be guarded. Thus we can generalise Lemma 9 to a result such that the resolvents of a query clause and a set of simple, weakly covering Horn clauses are query clauses.

Since a Horn LGC contains at most one positive literal, ordered factoring cannot be applied. Now we show other possibilities of applying resolution in *Query-Res*:

Lemma 10. *Using* Query-Res*, the resolvents of Horn LGCs are Horn LGCs.*

So far we showed that query clauses and Horn LGCs are closed under the inference system *Query-Res*, and the variable depth of derived clauses does not increase. Now we consider the length of derived clauses:

Lemma 11. *In an application of* Query-Res*, given a finite set of fixed Horn LGCs and fixed query clauses, all derived clauses are fixed query clauses.*

Now we give the second main result of this paper:

Theorem 5. Query-Res *decides the problem of the BCQ answering for Horn LGF.*

Restricted Queries for LGF

In this section, we answer loosely guarded queries, star queries and cloud queries over LGF. Theorem 4 implies that if a query clause Q is expressible in LGCs, then using *LGF-Res*, one can immediately answer Q over LGF. E.g., one can answer a loosely guarded query $\exists xyz(Postgrad(x) \wedge citedBy(x, y) \wedge citedBy(y, z) \wedge citedBy(z, x))$ over LGF using *LGF-Res*. This result is formally stated as:

Corollary 3. LGF-Res *decides the loosely guarded query answering problem for LGF.*

Another observation from Example 1 is: The top variable x does not occur with all other variables in Q. If y, which occurs with all other variables in Q, is a top variable, then using *Res*, the resolvent among Q, C_1, C_2 is an LGC. E.g., if we change C_1 to $C_1' = A_1(x_1, fx_1y_1) \vee B_1(gx_1y_1) \vee \neg G_1(x_1, y_1)$, to make y a top variable, then using *Res*, the resolvent of Q, C_1', C_2 is an LGC $B_1(gx_1y_1) \vee \neg G_1(x_1, y_1) \vee \neg G_2(x_1, y_1)$. This observation motivates our definition of star queries and cloud queries, which both guarantee the co-occurrence property between top variables and all other variables in the query clause.

Before discussing star/could queries, we first give the definition of partners of a variable. The *partner of a variable* x in a query Q $par(x, Q)$ is a set of variables that co-occur with x in an atom of Q, and $par(\overline{x}, Q)$ is interpreted as $par(x_1, Q) \cup \ldots \cup par(x_n, Q)$ where x_i is in an atom of Q. E.g., Let a query $Q = \exists xyz(A_1xy \wedge A_2yz)$. Then $par(x, Q) = \{y\}$ and $par(x, z, Q) = par(x, Q) \cup par(z, Q) = \{y\}$.

Definition 7 (Star Query)**.** *A BCQ is a* star query *Q if Q contains a top variable x such that $par(x, Q) = var(Q)$.*

The notion of star query strictly extends that of loosely guarded query, since only one top variable need to occur with all other variables. E.g., a query clause $Q = \exists xyz(A_1xy \wedge A_2yz)$ is not a loosely guarded query since x and z do not co-occur in any literal in Q, but if y is a top variable, Q is a star query.

Definition 8 (Cloud Query)**.** *A BCQ Q is a* cloud query *if Q satisfies these conditions:*

1. *Q contains chained variables $\mathcal{V}$ that are top variables.*
2. *Each pair in $\mathcal{V}$ co-occur in an atom of Q.*
3. *$par(\mathcal{V}, Q) = var(Q)$.*

The notion of cloud query is a further extension of that of a star query since a top variable does not need to occur with all other variables. An example of a cloud, but not star query is $Q = \exists xyzuv(A_1xy \wedge A_2yz \wedge A_3zuv \wedge A_4v)$ if y, z are top variables. Q is a cloud query because that $\mathcal{V} = \{y, z\}$, y, z co-occur in A_2, and $par(\mathcal{V}, Q) = var(Q)$.

Unlike loosely guarded queries, star queries and cloud queries vary depending on whether the top variables co-occur with all other variables.

Now we show that the resolvents of a set of LGCs and a star/cloud query are LGCs:

Lemma 12. *Using* Res *with refinement* LGC-Refine*, the resolvents of a set of LGCs and a star/cloud query are LGCs.*

Following the same idea of Lemma 8, we can show that the resolvents of a set of LGCs and a star/cloud query cannot have more variables than one of their side premises. Now we can state the third result of this paper:

Theorem 6. Query-Res *decides the problem of the loosely guarded query and star/cloud query answering for LGF.*

Conclusion and Future Work

In this paper, we have presented, as far as we know, the first practical decision procedures for answering BCQs over Horn LGF. Inspired by the 'MAXVAR' notion from (de Nivelle and de Rijke 2003), we used the top variable technique to handle chained variables in query clauses. Based on this top variable technique, we showed the method *LGF-Res* decides LGF, the method *Query-Res* answers BCQs over Horn LGF, and it answers loosely guarded queries, star queries and cloud queries over LGF. This shows that *Query-Res* provides essentials for the implementation of query answering over guard-related fragments as an extension for existing first-order logic reasoners.

Using *Query-Res*, an issue of answering BCQs against the whole of LGF is that: If the top variable is an isolated variable in a query clause Q, then the resolvents of Q and a set of LGCs is neither a query clause nor an LGC (Example 1). Our next step is extending *Query-Res* so that we can use it to answer BCQs against LGF, for which as yet there is no practical procedure.

Acknowledgements

We very thank the anonymous reviewers for their useful comments.

References

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge Univ. Press.

Bachmair, L., and Ganzinger, H. 2001. Resolution theorem proving. In Robinson, A., and Voronkov, A., eds., *Handbook of Automated Reasoning*. Elsevier and MIT Press. 19–99.

Bachmair, L.; Ganzinger, H.; and Waldmann, U. 1993. Superposition with simplification as a decision procedure for the monadic class with equality. In *Proc. Computational Logic and Proof Theory*, 83–96. Springer.

Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2011. On rules with existential variables: Walking the decidability line. *Artif. Int.* 175(9):1620–1654.

Bárány, V.; Gottlob, G.; and Otto, M. 2010. Querying the guarded fragment. In *Proc. LICS'10*, 1–10. IEEE Computer Society.

Bárány, V.; ten Cate, B.; and Segoufin, L. 2015. Guarded negation. *J. ACM* 62(3):22:1–22:26.

Bernstein, P. A., and Chiu, D.-M. W. 1981. Using semi-joins to solve relational queries. *J. ACM* 28(1):25–40.

Calì, A.; Gottlob, G.; and Kifer, M. 2013. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Int. Res.* 48(1):115–174.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general datalog-based framework for tractable query answering over ontologies. *J. Web Semantics* 14:57–83.

de Nivelle, H., and de Rijke, M. 2003. Deciding the guarded fragments by resolution. *J. Symb. Comput.* 35(1):21–58.

Dershowitz, N. 1982. Orderings for term-rewriting systems. *Theoretical Comp. Sci.* 17(3):279–301.

Fermüller, C.; Leitsch, A.; Tammet, T.; and Zamov, N. 1993. *Resolution Methods for the Decision Problem*, volume 679 of *LNAI*. Springer.

Ganzinger, H., and de Nivelle, H. 1999. A superposition decision procedure for the guarded fragment with equality. In *Proc. LICS'99*, 295–303. IEEE.

Ganzinger, H.; Hustadt, U.; Meyer, C.; and Schmidt, R. A. 1998. A resolution-based decision procedure for extensions of K4. In *Proc. AiML'98*, 225–246. CSLI.

Glimm, B.; Lutz, C.; Horrocks, I.; and Sattler, U. 2008. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. *J. Artif. Int. Res.* 31(1):157–204.

Gottlob, G.; Leone, N.; and Scarcello, F. 2003. Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *J. Comp. and Syst. Sci.* 66(4):775–808.

Gottlob, G.; Pieris, A.; and Tendera, L. 2013. Querying the guarded fragment with transitivity. In *Proc. of ICALP'13*, 287–298. Springer.

Grädel, E. 1999. On the restraining power of guards. *J. Symb. Logic* 64(4):1719–1742.

Hustadt, U., and Schmidt, R. A. 1997. On evaluating decision procedures for modal logic. In *Proc. IJCAI'97*, 202–207. Morgan Kaufmann.

Hustadt, U., and Schmidt, R. A. 1999. Maslov's class K revisited. In *Proc. CADE'99*, LNAI, 172–186. Springer.

Hustadt, U. 1999. *Resolution based decision procedures for subclasses of first-order logic*. Ph.D. Dissertation, Saarland Univ., Saarbrücken, Germany.

Nonnengart, A., and Weidenbach, C. 2001. Computing small clause normal forms. In Robinson, A., and Voronkov, A., eds., *Handbook of Automated Reasoning*. Elsevier and MIT Press. 335–367.

Riazanov, A., and Voronkov, A. 2001. Vampire 1.1 (system description). In *Proc. IJCAR'01*, LNCS, 376–380. Springer.

Schulz, S. 2013. System Description: E 1.8. In *Proc. LPAR'13*, LNCS. Springer.

van Benthem, J. 1997. Dynamic bits and pieces. Research Report LP-97-01, Univ. of Amsterdam.

Vardi, M. Y. 2000. Constraint satisfaction and database theory: A tutorial. In *Proc. PODS'00*, 76–85. ACM.

Weidenbach, C.; Dimova, D.; Fietzke, A.; Kumar, R.; Suda, M.; and Wischnewski, P. 2009. Spass version 3.5. In *Proc. CADE'09*, 140–145. Springer.

Xiao, G.; Calvanese, D.; Kontchakov, R.; Lembo, D.; Poggi, A.; Rosati, R.; and Zakharyaschev, M. 2018. Ontology-based data access: A survey. In *Proc. IJCAI'18*, 5511–5519. IJCAI.