J. Symbolic Computation (2004) **11**, 1–000

Deciding the Guarded Fragments by Resolution

Hans de Nivelle[1] and Maarten de Rijke[1]
[1] ILLC, University of Amsterdam
Plantage Muidergracht 24, 1018 TV Amsterdam
`mdr@wins.uva.nl, nivelle@mpi-sb.mpg.de`

(Received 2 March 2004)

The guarded fragment is a fragment of first-order logic that has been introduced for two main reasons: First, to explain the good computational and logical behavior of propositional modal logics. Second, to serve as a breeding ground for well-behaved process logics. In this paper we give resolution-based decision procedures for the guarded fragment and for the loosely guarded fragment (sometimes also called pairwise guarded fragment). By constructing an implementable decision procedure for the guarded fragment and for the loosely guarded fragment, we obtain an effective procedure for deciding modal logics that can be embedded into these fragments. The procedures have been implemented in the theorem prover `Bliksem`.

1. Introduction

The guarded fragment was inspired by two observations. First, many propositional modal logics have very good computational and logical properties: their satisfiability problems are decidable in polynomial space and exponential time; they have the (uniform) finite model property, and the tree model property (Vardi, 1997); we have a solid understanding of their expressive power in model theoretic terms, and they have various interpolation and preservation properties. See (de Rijke, 1999).
Second, these modal logics can be translated into first-order logic, using a standard (relational) translation based on the Kripke semantics. In this translation, a modal formula A is translated by computing $T(A, x, y)$, where x and y are two distinct first-order variables. T is recursively defined as follows:

$$
\begin{array}{lcl}
T(p,\alpha,\beta) & = & p(\alpha) \text{ if } p \text{ is an atom} \\
T(\neg A,\alpha,\beta) & = & \neg T(A,\alpha,\beta) \\
T(A \vee B,\alpha,\beta) & = & T(A,\alpha,\beta) \vee T(B,\alpha,\beta) \\
T(A \wedge B,\alpha,\beta) & = & T(A,\alpha,\beta) \wedge T(B,\alpha,\beta) \\
T(A \rightarrow B,\alpha,\beta) & = & T(A,\alpha,\beta) \rightarrow T(B,\alpha,\beta) \\
T(\Box A,\alpha,\beta) & = & \forall\beta[R(\alpha,\beta) \rightarrow T(A,\beta,\alpha)] \\
T(\Diamond A,\alpha,\beta) & = & \exists\beta[R(\alpha,\beta) \wedge T(A,\beta,\alpha)]
\end{array}
$$

Here R is a binary relation symbol that denotes the accessibility relation. In case there are additional restrictions on the accessibility relation, these can be explicitly added to

0747–7171/90/000000 + 00 $03.00/0

the translation. The formula $T(A,x,y)$ means 'A holds in world x '. In order to translate 'A is satisfiable', one must compute $\exists x T(A,x,y)$.
The consequence of the translation above is that propositional modal logics can be seen as fragments of first-order logic. The natural question that arises is: What makes these fragments special? Or put differently, why do they have the pleasant computational and logical properties noted above? Gabbay in (Gabbay, 1981) was the first to observe that modal logics can be translated into the *2-variable fragment* FO^2 of first-order logic, which is decidable. (Indeed the translation given above uses only the variables x and y) The fragment FO^2 with equality was first shown to be decidable in (Mortimer, 1975), without giving an explicit complexity bound. In (Grädel *et al.*, 1997) it was shown that the satisfiability problem for the 2-variable fragment (with equality) is NEXPTIME-complete. In (Grädel *et al.*, 1997) an interesting account of the history of the fragment can be found.

The decidability of FO^2 appears to be an explanation for the pleasant properties of modal logics. We have a clear understanding of the expressive power of FO^2 in terms of so-called pebble games (Immerman and Kozen, 1989). However on the negative side, FO^2 is not finitely axiomatizable, it does not have the Craig interpolation property, and it does not have the tree model property, unlike the modal logics it contains. For example, the formula $\forall xy[R(x,y)]$ does not have a tree-like model. In (Vardi, 1997) it is convincingly argued that the tree model property is the reason for the good behavior of modal logics. Recently, an alternative explanation for the good behavior of modal fragments of first-order logic was put forward by Andréka *et al.* in 1998. Their observation is that in the translation given above all quantifiers only occur *relativized* or *guarded* by the accessibility relation. They called this fragment of first-order logic, in which all quantifiers occur relativized, the *guarded fragment.* Clearly, the translation above translates modal formulae into the guarded fragment.
At present, the guarded fragment is actively being investigated, both from a computational and from a logical point of view. It is known to be decidable and to have the finite model property (Andréka *et al.*, 1998). Its satisfiability problem is decidable in double exponential time and it enjoys (a generalized form of) the tree model property (Grädel, 1997). Because of this it is consistent with (Vardi, 1997) to use the guarded fragment as explanation for the good behavior of modal logics.
Actually, the results in (Grädel, 1997) were proven for the guarded fragment with equality. (however equality cannot act as a guard) In (Ganzinger *et al.*, 1999) it is shown that the 2-variable restriction of the guarded fragment remains decidable, when it is extended by transitive relations. In (Grädel and Walukiewicz, 1999), the guarded fragment is extended with monotone fixed point constructors. It is shown that this extension does not increase the complexity of the decision problem. Moreover, this extended fragment still satisfies the tree model property.
Many familiar—and well-behaved—modal logics can be translated into the guarded fragment. These logics include K, B, D, and recently also $S4, K4$ and $S5$ (de Nivelle, 1999b). However, it seems that several important modal and temporal logics can not be translated into the guarded fragment, including the temporal logic with Since and Until. For these reasons, a number of generalizations of the guarded fragment have been proposed, the oldest of which is the so-called *loosely guarded fragment* (van Benthem, 1997). In this fragment, more liberal guards are allowed than in the original guarded fragment. With these liberal guards the operators Since and Until can be translated.
The aim of this paper is to present resolution decision procedures for both the guarded

fragment and the loosely guarded fragment without equality. Recently, a superposition decision procedure for the guarded fragment with equality has been developed in (Ganzinger and de Nivelle, 1999). Although the first-order fragment in that paper is more general, the clause fragment had to be strongly restricted in order to make it possible to include equality. For example, the clause fragment used here allows nesting of function symbols, while this is not allowed in the other clause fragment. This makes that the decidability results here and the decidability results in (Ganzinger and de Nivelle, 1999) are incomparable at the clause level.

In order to decide the guarded fragment, we define guarded clauses, and show that first-order guarded formulae can be translated into sets of guarded clauses. After that we show that sets of guarded clause sets are decidable by an appropriate restriction of resolution. The restriction that has to be used is based on a so-called *ordering refinement*. All of the resolution theorem provers (SPASS (Weidenbach, 1997), OTTER (McCune, 1995), and Bliksem (de Nivelle, 1999a)) support orderings. This makes that our strategy fits very well into the standard framework of first-order resolution theorem proving. The standard optimizations and implementation techniques can be reused for our decision procedure, so we can expect our procedure to be technically efficient. Indeed, with an effective resolution-based decision procedure, implementation has become feasible. The strategy for the guarded fragment has been implemented in the theorem prover Bliksem (de Nivelle, 1999a). We will also show that our decision procedure is theoretically optimal, because it terminates in double exponential time.

In order to decide the loosely guarded fragment we define a similar notion of loosely guarded clause. However, deciding sets of loosely guarded clauses is much harder than deciding sets of guarded clauses. We need a non-trivial modification of hyperresolution on top of the ordering refinement for this. In order to prove its completeness, an extension of the resolution game turns out to be necessary.

The paper is organized as follows. Section 2 provides background material. After that, in Section 3 we get to work and establish decidability of the guarded fragment by means of ordered resolution. In Section 4 we use ordered resolution to decide the loosely guarded fragment. The fifth and final section contains our conclusions and as well as some open questions.

2. Background

We begin by defining the guarded fragment. After that we give some general background on resolution strategies, normal from transformations, and covering literals. It should be noted that we do not consider equality in this paper. For this we refer to (Ganzinger and de Nivelle, 1999).

2.1. The Guarded Fragment

DEFINITION 2.1. *The* guarded fragment *(GF) is recursively defined as the following subset of first-order logic without equality and function symbols.*

1 $\top$ *and* $\bot$ *are in GF.*
2 If a *is an atomic formula, then* $a \in \mathrm{GF}$.
3 If $A, B \in \mathrm{GF}$, *then* $\neg A,\ A \vee B,\ A \wedge B,\ A \rightarrow B,\ A \leftrightarrow B \in \mathrm{GF}$.

4 *Let* $A \in \mathrm{GF}$, *and let* a *be an atomic formula such that every free variable of* A *occurs at least once among the arguments of* a. *Then* $\forall\overline{x}(a \rightarrow A) \in \mathrm{GF}$ *and* $\exists\overline{x}(a \wedge A) \in \mathrm{GF}$. *We also allow* $\forall\overline{x}(\neg a \vee A) \in \mathrm{GF}$.

The atoms a *in Item 4 are called* guards.

There are no conditions on the order in which the variables occur in the guards. It is also allowed to repeat variables.

EXAMPLE 2.2. *The following formulae are guarded:*

$$\forall xy[a(x,y) \rightarrow (b(x,y) \wedge c(x) \wedge d(y,y))].$$

$$\forall xy[a(x,y,y,x) \wedge (c(x) \vee \neg\forall z[a(y,z) \rightarrow d(y)])].$$

The following formulae are not guarded:

$$\forall xy[a(x) \rightarrow a(f(x))].$$

$$\forall xy(a(x) \rightarrow b(x,y)).$$

$$\forall xyz[R(x,y) \wedge R(y,z) \rightarrow R(x,z)].$$

It is easily checked that for every modal formula A the translation $\exists x T(A,x,y)$ is guarded. $T(A,x,y)$ is clearly function free. The set of free variables of $T(B,\alpha,\beta)$ is always included in $\{\alpha\}$. All quantifications in $T(A,x,y)$ have form $\forall\beta[R(\alpha,\beta) \rightarrow T(B,\beta,\alpha)]$ or $\exists\beta[R(\alpha,\beta) \wedge T(B,\beta,\alpha)]$. Since β occurs in $R(\alpha,\beta)$ the quantifications are guarded.

2.2. RESOLUTION

We briefly review some elementary facts about resolution. We assume that the reader is familiar with such notions as literals, clauses, and ground terms. We begin by defining some complexity measures for terms, atoms, clauses, and literals. For convenience we identify atoms and terms in the following recursive definitions. Let A be an atom/term. The *depth* of A is recursively defined as follows:

1 If A is a variable, then $\mathrm{Depth}(A) = 1$.
2 For a functional term/atom, $\mathrm{Depth}(f(t_1,\ldots,t_n))$ equals the maximum of $\{1, 1 + \mathrm{Depth}(t_1), \ldots, 1 + \mathrm{Depth}(t_n)\}$.

The depth of a literal equals the depth of its atom. The depth of a clause c equals the maximal depth of the literals in c, or 0 for the empty clause.
The *vardepth* of a term/atom A is recursively defined as follows:

1 If A is ground, then $\mathrm{Vardepth}(A) = -1$.
2 If A is a variable, then $\mathrm{Vardepth}(A) = 0$.
3 In all other cases,

$$\mathrm{Vardepth}(f(t_1,\ldots,t_n)) = \max\{1 + \mathrm{Vardepth}(t_1), \ldots, 1 + \mathrm{Vardepth}(t_n)\}.$$

The vardepth of a literal equals the vardepth of its atom. The vardepth of a clause c equals the maximal vardepth of a literal in c. The vardepth of the empty clause is defined as -1.
If A is an atom, literal, or clause, then $\mathrm{Var}(A)$ is defined as the set of variables that occur in A. $\mathrm{Varnr}(A)$ is defined the number of variables in A, i.e. as the cardinality of $\mathrm{Var}(A)$. For a term/atom A, we define the *complexity* of A, written as $\#A$, as the total number of occurrences of function, constant, and variable symbols in A.

Next we introduce the ordered resolution rule. We assume that the reader is familiar with most general unifiers (mgu's); see (Chang and Lee, 1973) or (Leitsch, 1997).

DEFINITION 2.3. *We define the ordered resolution rule, and factorization rule. Let $\sqsubset$ be an order on literals.*

Res *Let $\{A_1\} \cup R_1$ and $\{\neg A_2\} \cup R_2$ be two clauses such that the following hold:*

1 $\{A_1\} \cup R_1$ and $\{\neg A_2\} \cup R_2$ have no variables in common;
2 there is no $A \in R_1$ such that $A_1 \sqsubset A$;
3 there is no $A \in R_2$ such that $A_2 \sqsubset A$; and
4 A_1 and A_2 have an mgu Θ.

Then the clause $R_1\Theta \cup R_2\Theta$ is called a $\sqsubset$-ordered resolvent *of $\{A_1\} \cup R_1$ and $\{\neg A_2\} \cup R_2$.*

Fact *Let $\{A_1, A_2\} \cup R$ be a clause, such that*

1 there is no $A \in R$ such that $A_1 \sqsubset A$;
2 A_1 and A_2 have an mgu Θ.

Then the clause $\{A_1\Theta\} \cup R\Theta$ is called a $\sqsubset$-ordered factor *of $\{A_1, A_2\} \cup R$.*

The order $\sqsubset$ is called *liftable* if it satisfies the following condition, for all literals A, B, and for all substitutions Θ,

$$A \sqsubset B \Rightarrow A\Theta \sqsubseteq B\Theta.$$

The combination of ordered resolution and factoring is complete, when the order is liftable, see (Leitsch, 1997) for a proof. The order that we will use for the guarded fragment does not satisfy this property.
We now define (unordered) hyperresolution. We mention hyperresolution here because we will need a variant of it in the decision procedure for the loosely guarded fragment.

DEFINITION 2.4. *Let $\{A_1\} \cup R_1, \ldots, \{A_p\} \cup R_p$ be purely positive clauses. Let $\{\neg A'_1, \ldots, \neg A'_p\} \cup \{B_1, \ldots, B_q\}$ be a mixed clause, in which $B_1, \ldots, B_q$ are positive. Let Θ be the most general unifier of the pairs*

$$(A_1, A'_1), \ldots, (A_p, A'_p).$$

Then the clause

$$R_1\Theta \cup \cdots \cup R_p\Theta \cup \{B_1\Theta, \ldots, B_q\Theta\}$$

is a hyperresolvent.

2.3. Transformation to Clausal Normal Form

Resolution works only on formulae of a restricted form. In order to be able to deal with full first-order logic, we need a method of transforming first-order formulae into clause sets. We give a collection of operators that can be used for this transformation. We define all operators as working on sets of formulae rather than on formulae themselves, so that operators can split one formula into different formulae. To start, here is a brief overview:

NNF(C) Bring C in negation normal form.
Struct(C) Replace certain subformulae by fresh atoms, and add equivalence definitions for the new atoms.
Struct$_+$(C) Replace certain subformulae by fresh atoms, but add implications instead of equivalences.
Sk(C) Replace every existentially quantified variable by a functional term, using a fresh function symbol.
Cls(C) Factor C into a set of clauses.

The operator sequence NNF, Sk, Cls constitutes a complete transformation. It is possible to insert Struct or Struct$_+$ before Cls.

Definition 2.5. *Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae.* NNF(C) *is obtained by first replacing all occurrences of $\rightarrow$ and $\leftrightarrow$, after that moving all $\neg$'s inwards as much as possible, and by finally removing all double $\neg$'s.*

In (Baaz *et al.*, 1994) the *structural transformation* is defined by replacing all subformulae of a certain formula by fresh names, with defining formulae for the fresh names. When such a tranformation has been applied, the original formula can always be reconstructed, contrary to when the normal form has been obtained by factoring. For this reason Baaz, Fermüller and Leitsch have called these transformations *structural*. In our decision procedures we will make use of structural transformations, but we will not replace all subformulae. We will now give the operator Struct but specify later which subformulae are going to be replaced.

Definition 2.6. *Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae. We define* Struct(C) *as the result of making replacements of the following form: Let A be a subformula of one of the F_i. Let $x_1, \ldots, x_n$ be an enumeration of the free variables of A. Let α be a new predicate name. Replace $F_i[A]$ by $F_i[\alpha(x_1, \ldots, x_n)]$ and add*

$$\forall x_1, \ldots, x_n \; [\alpha(x_1, \ldots, x_n) \leftrightarrow A]$$

to C.
If C is in negation normal form, then it is sufficient to use $\rightarrow$ instead of $\leftrightarrow$ in order to obtain a satisfiability preserving transformation. Struct$_+$ *is defined by adding $\forall x_1, \ldots, x_n \; [\alpha(x_1, \ldots, x_n) \rightarrow A]$ to C, instead of using equivalence.*

Definition 2.7. *Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae in negation normal form. We define the* Skolemization Sk(C) *as the result of making the following replacements: As long as one of the F_i contains an existential quantifier, write $F_i = F_i[\exists y \; A]$, where*

$\exists y\ A$ is not in the scope of another existential quantifier. Let $x_1, \ldots, x_n$ be the universally quantified variables in the scope of which A occurs. Replace $F_i[\exists y\ A]$ by $F_i[A[y := f(x_1, \ldots, x_n)]\ ]$. Here we use the notation $F_i[y := t]$ to denote full first-order substitution.

There are more sophisticated ways for Skolemization leading to more general Skolem terms, see (Ohlbach and Weidenbach, 1995), but we cannot use them for our present purposes.

DEFINITION 2.8. *Let $C = \{F_1, \ldots, F_n\}$ be a set of formulae in NNF containing no existential quantifiers: The* clausification *of C, written as* Cls(C)*, is the result of the following replacements.*

1. *Replace $A \vee (B \wedge C)$ by $(A \vee B) \wedge (A \vee C)$.*
2. *Replace $(A \wedge B) \vee C$ by $(A \vee C) \wedge (B \vee C)$.*
3. *Replace $\forall x\, A$ by $A[x := X]$, where X is a designated variable symbol not occurring in A.*
4. *If one of the F_i has form $A \wedge B$, then replace it by A and B.*

The result of Cls is of a set of clauses.

2.4. Weakly Covering Literals

In this section we briefly introduce a class of literals that are called *weakly covering literals*. They first appeared in (Tammet, 1990), and independently in the thesis of Fermüller, see (Fermüller *et al.*, 1993). Weakly covering literals are the basis of many of the classes that are decidable by resolution, such as E^+ and S^+. Their usefulness is due to the fact that when two weakly covering literals are unified, the result is not more complex than the larger of them. We will shortly state the main facts.

DEFINITION 2.9. *A literal is* covering *if every functional subterm of it contains all variables that occur in the literal. A literal is* weakly covering *if every non-ground, functional subterm contains all variables of the literal.*

We will not make use of covering literals, but we included the definition for the sake of completeness. Covering and weakly covering literals are typically the result of Skolemization, when the prefix ends in an existential quantifier. If a function free atom $a(\overline{x}, y)$ in the scope of quantifiers $\forall\overline{x}\exists y$ is Skolemized, the result equals $a(\overline{x}, f(\overline{x}))$, which is covering. If $a(\overline{x}, y)$ contains functional ground terms, then the result is weakly covering. For the proofs of the following facts we refer to (Fermüller *et al.*, 1993). We mention the facts here so that we can refer to them when we need them in later sections.

THEOREM 2.10. *Let A and B be weakly covering literals that have an mgu Θ. Let $C = A\Theta = B\Theta$. Then*

1. *C is weakly covering.*
2. *One of the following holds: Either* $\mathrm{Vardepth}(C) \leq \mathrm{Vardepth}(A)$ *and* $\mathrm{Varnr}(C) \leq \mathrm{Varnr}(A)$, *or* $\mathrm{Vardepth}(C) \leq \mathrm{Vardepth}(B)$ *and* $\mathrm{Varnr}(C) \leq \mathrm{Varnr}(B)$.

Theorem 2.10 alone does not prevent unbounded growth of the unifier. This is because of the fact that, although the variable depth of C is bounded, C may contain arbitrarily large ground terms. The following controls this problem:

LEMMA 2.11. *Let $C = A\Theta = B\Theta$ be a most general unifier of two weakly covering literals. Let v be the maximum of* $\text{Vardepth}(A)$ *and* $\text{Vardepth}(B)$. *Every ground term in C occurring at a depth greater than or equal to v, occurs either in A or in B.*

This restricts the introduction of new ground terms to ground clauses. This will turn out sufficient for bounding the growth of unified terms.
What we have until now is not sufficient for bounding the side literals in resolved clauses. Let $R_1\Theta \cup R_2\Theta$ be a resolvent of $\{A_1\} \cup R_1$ and $\{\neg A_2\} \cup R_2$. Theorem 2.10 states that $A_1\Theta$ is weakly covering and bounded in variable depth, but we have said nothing about the literals in $R_i\Theta$. First we state that the side literals are weakly covering, after that we state that their variable depth is bounded.

THEOREM 2.12. *Let A and B be literals which are both weakly covering. Let* $\text{Var}(A) \subseteq \text{Var}(B)$, *and let Θ be a substitution such that $B\Theta$ is weakly covering. Then $A\Theta$ is weakly covering.*

LEMMA 2.13. *Let A and B literals, which are both weakly covering. Let* $\text{Var}(A) \subseteq \text{Var}(B)$, $\text{Vardepth}(A) \leq \text{Vardepth}(B)$, *and let Θ be a substitution. Then* $\text{Vardepth}(A\Theta) \leq \text{Vardepth}(B\Theta)$, *and* $\text{Var}(A\Theta) \subseteq \text{Var}(B\Theta)$.

2.5. The Resolution Game

The completeness proof of our strategy is based on the resolution game, which was introduced in (de Nivelle, 1994) as a device for proving completeness of resolution with non-liftable orders. We briefly introduce it here, but for a more elaborate description, see (de Nivelle, 1994).

DEFINITION 2.14. *A* resolution game *is an ordered triple $\mathcal{G} = (P, \mathcal{A}, \prec)$, where*

1 P is a set of propositional symbols,
2 $\mathcal{A}$ is a set of attributes,
3 $\prec$ is an order on $(P \cup \neg P) \times \mathcal{A}$, where $\neg P$ is defined as $\{\neg p \mid p \in P\}$.

It must be the case that $\prec$ is well-founded on $(P\cup\neg P)\times\mathcal{A}$. The elements of $(P\cup\neg P)\times\mathcal{A}$, are called indexed literals. *We will write $a{:}A$ instead of (a, A). A* clause *of $\mathcal{G}$ is a finite multiset of indexed literals of $\mathcal{G}$.*

Interpretations for a resolution game are defined in a standard manner, i.e., as propositional assignments. A clause is true in an interpretation if one of the literals that occurs in it (ignoring the indices) is true. We now define resolution and factoring for the resolution game. We need an explicit factoring rule even for propositional logic, because clauses are multisets.

DEFINITION 2.15. *Let $\mathcal{G} = (P, \mathcal{A}, \prec)$ be a resolution game. Let c be a clause of $\mathcal{G}$. An* *indexed literal* $a{:}A$ *is* maximal *in* c, *if for no indexed literal* $b{:}B$ *in* c, $a{:}A \prec b{:}B$. *We define resolution and factoring for* $\mathcal{G}$: *Let* $c_1 = [a{:}A_1] \cup r_1{:}R_1$ *and* $c_2 = [\neg a{:}A_2] \cup r_2{:}R_2$ *be clauses such that* $a{:}A_1$ *and* $\neg a{:}A_2$ *are maximal in their clauses. Then* $r_1{:}R_1 \cup r_2{:}R_2$ *is a* resolvent *of* c_1 *and* c_2. *The expressions* $r_i{:}R_i$ *denote finite multisets of indexed literals. Let* $c_1 = [a{:}A_1, a{:}A_2] \cup r{:}R$ *be a clause, such that* $a{:}A_1$ *is maximal in* c_1. *Then* $[a{:}A_1] \cup r{:}R$ *is a* factor *of* c_1.

Until now we have nothing unusual, as this is just lock resolution (Boyer, 1971). We now define reductions, which distinguish the resolution game from lock resolution.

Definition 2.16. *Let* c *be a clause of a resolution game* $\mathcal{G}$. *A* reduction *of* c *is obtained by performing zero, or any finite number of the following actions:* **(1)** *Deleting an indexed literal.* **(2)** *Replacing an indexed literal* $a{:}A_1$ *by an indexed literal* $a{:}A_2$ *with* $a{:}A_2 \prec a{:}A_1$.

Definition 2.17. *Let* C *be a set of clauses of a resolution game* $\mathcal{G} = (P, \mathcal{A}, \prec)$. *A* saturation $\overline{C}$ *of* C *is a minimal set for which* **(1)** $C \subseteq \overline{C}$. **(2)** *For every resolvent* c *that can be constructed from two clauses* $c_1, c_2 \in \overline{C}$, *there is a reduction* d *of* c *in* $\overline{C}$. **(3)** *For every factor* c *that can be constructed from a clause* $c_1 \in \overline{C}$, *there is a reduction* d *of* c *in* $\overline{C}$.

The resolution game is different from lock or indexed resolution (Boyer, 1971), because in lock resolution the resolvent inherits the indices from the parent clause without any changes. In the resolution game the indices may change. The reason that this variant of resolution is called resolution game, is that it can be seen as a game of two players: One player, called the *opponent*, is trying to refute the clause set using ordered resolution and factoring. The other player, called the *defender*, tries to disturb the opponent by replacing clauses by reductions.

Theorem 2.18. *Let* C *be a set of clauses of a resolution game* $\mathcal{G}$. *The following two statements are equivalent:* **(1)** C *is unsatisfiable.* **(2)** *Every saturation of* C *contains the empty clause.*

A complete proof can be found in (de Nivelle, 1994). In terms of games, Theorem 2.18 can be reformulated as follows: If C is unsatisfiable, then the opponent has a winning strategy, and if C is satisfiable, then the defender has a winning strategy.

3. The Guarded Fragment

In this section we give a decision procedure for the guarded fragment. Our decision procedure is based on ordered resolution, as defined in Definition 2.3. It is common to restrict the resolution rule by an ordering, but usually this is done to improve efficiency in cases where a proof exists. However, certain orderings can be used to enforce termination in cases where no proof exists.

We will illustrate this point with an example. Let C be some clause set in which only one variable X is used, all literals contain this variable X, and which contains no constant symbols. So $\{p(X), q(s(X,X),X)\}$ is allowed, but $\{p(s(X),0)\}$ is not. Let $\sqsubset$ be an order on literals that is defined by putting $A \sqsubset B$ iff $\text{Vardepth}(A) < \text{Vardepth}(B)$. Then the following hold:

1 Every ordered resolvent or factor from C contains exactly one variable, and no constants. Hence every derivable clause can be renamed such that it contains only the variable X.
2 If Θ is the mgu of two literals A and B, each containing exactly one variable and no constant symbol, then $A\Theta$ and $B\Theta$ are also such literals, and $\text{Vardepth}(A\Theta) = \text{Vardepth}(B\Theta)$ is equal to $\text{Vardepth}(A)$ or to $\text{Vardepth}(B)$.
3 If $\text{Vardepth}(A) < \text{Vardepth}(B)$, and $\Theta = \{X := t\}$ is a substitution, such that t contains exactly one variable and no constants, then $\text{Vardepth}(A\Theta) < \text{Vardepth}(B\Theta)$.

As a consequence, the clauses cannot become deeper, and cannot contain more than one variable. Because the set of literals that can occur in the clauses is finite, the set of derivable clauses is finite. Hence, the order $\sqsubset$ enforces termination. If one can show the completeness of resolution with $\sqsubset$, at least for this one-variable class, then one has a decision procedure. This is straightforward because the order is liftable on the class under consideration. Our decidability proofs below have the same structure as this example.

3.1. Basics

In order to be able to use resolution we need a notion of guardedness for clause sets, and a way to translate guarded, first-order formulae into guarded clause sets. The translation is not completely standard. Standard translations would transform guarded formulae into non-guarded clauses.
The first step of the transformation is the transformation into NNF. This can be done without problems, since all of the necessary replacements preserve the guarded fragment. When the formula is in NNF, the guard condition for the existential quantifier is not necessary anymore. This means that the guard condition in Definition 2.1 can be weakened to positively occurring $\forall$-quantifiers, and negatively occurring $\exists$-quantifiers, in the case where one wants to decide satisfiability. For clause sets we define the following normal form.

Definition 3.1. *A clause c is called* guarded *if it satisfies the following conditions:*

1 Every non-ground, functional term in c contains all variables of c.
2 If c is not ground, then there is a negative literal $\neg A$ in c that does not contain a non-ground, functional term, and that contains all variables of c.

A clause set C is called guarded *if its clauses are guarded.*

The negative literal in item 2 of Definition 3.1 is the guard. Every ground clause is guarded. The definition of a guarded clause given here differs from the definition in (de Nivelle, 1998) but is equivalent. In (de Nivelle, 1998) the first condition was given as two conditions: **(1a)** Every literal, containing non-ground functional terms contains all variables of c, and **(1b)** every literal in c is weakly covering. It is easily checked that **1a** and **1b** is equivalent with **(1)**.

Example 3.2. The clause $\{p(0, s(0)), q(s(s(0)))\}$ is guarded because it is ground. The clause $\{\neg p(X), \neg q(X,Y), r(f(X,Y))\}$ is guarded by the literal $\neg q(X,Y)$. The clause $\{\neg p(X), \neg q(Y), r(f(X,Y))\}$ is not guarded. Adding a literal $\neg a(X,Y,X,X,Y)$ would result in a guarded clause. The clause $\{\neg p(Y,X), q(f(X),X,Y)\}$ is not guarded. It cannot be made guarded by adding literals. The empty clause is guarded.

Let us continue with the translation taking guarded formulae into guarded clause sets. We need a variant of Struct_+ of Definition 2.6, which we will call $\mathrm{Struct}_\forall$.

DEFINITION 3.3. $\mathrm{Struct}_\forall$ *is the structural transformation that is obtained by replacing the subformulae of the forms* $\forall\overline{x}(a \to A)$ *or* $\forall\overline{x}(\neg a \vee A)$ *with free variables* $\overline{y}$, *by some fresh name* $\alpha(\overline{y})$ *and adding a defining formula of the form* $\forall\overline{xy}\ (\neg a \vee \neg\alpha \vee A)$. *The latter formula is equivalent with* $\forall\overline{y}(\alpha \to \forall\overline{x}(a \to A))$.

EXAMPLE 3.4. The guarded formula

$$\exists x\, n(x) \wedge \forall y\, [a(x,y) \to \neg\exists z(p(x,z) \wedge (\forall x\ a(x,z) \to (b(z,z) \wedge c(x,x))))]$$

is translated as follows. First, NNF results in

$$\exists x\, n(x) \wedge \forall y\, [\neg a(x,y) \vee \forall z(\neg p(x,z) \vee (\exists x\ a(x,z) \wedge (\neg b(z,z) \vee \neg c(x,x))\ ))].$$

After that, $\mathrm{Struct}_\forall$ results in the following set of formulae

$$\exists x\, [n(x) \wedge \alpha(x)], \quad \forall xy\, [\neg a(x,y) \vee \neg\alpha(x) \vee \beta(x)],$$
$$\forall xz\, [\neg p(x,z) \vee \neg\beta(x) \vee (\exists x\ a(x,z) \wedge (\neg b(z,z) \vee \neg c(x,x)))].$$

Sk results in

$$n(c) \wedge \alpha(c), \quad \forall xy\, [\neg a(x,y) \vee \neg\alpha(x) \vee \beta(x)]$$
$$\forall xz\, [\neg p(x,z) \vee \neg\beta(x) \vee (a(f(x,z),z) \wedge (\neg b(z,z) \vee \neg c(f(x,z),f(x,z))\ ))].$$

And finally, clausification results in

$$\{n(c)\}, \quad \{\alpha(c)\}, \quad \{\neg a(X,Y), \neg\alpha(X), \beta(X)\},$$
$$\{\neg p(X,Z), \neg\beta(X), a(f(X,Z),Z)\},$$
$$\{\neg p(X,Z), \neg\beta(X), \neg b(Z,Z), \neg c(f(X,Z),f(X,Z))\}.$$

THEOREM 3.5. *Let* $F \in \mathrm{GF}$. *Then*

1 $F' = \mathrm{NNF}(F) \in \mathrm{GF}$,
2 $F'' = \mathrm{Struct}_\forall(F') \in \mathrm{GF}$, *and*
3 $(\mathrm{Sk};\mathrm{Cls})(F'')$ *is a guarded clause set.*

PROOF. We consider the steps made in the transformation: The NNF is characterized by a set of rewrite rules. Let $\Phi = \forall\overline{x}\ (a \to A)$ or $\Phi = \exists\overline{x}\,(a \wedge A)$ be a guarded quantification. Φ will remain guarded under each application of a rewrite rule inside A, since none of the rewrite rules introduces a free variable. Similarly if Φ occurs in the X or Y of a rewrite rule $(X \text{ op } Y) \Rightarrow \cdots$ then A is copied without problems. The only possible problem occurs when $\forall\overline{x}\,(a \to A)$ rewrites to $\forall\overline{x}\,(\neg a \vee A)$, but this case is covered by the definition of the guarded fragment.
The next step is $\mathrm{Struct}_\forall$. The defining formula $\forall\overline{xy}\ (\neg a \vee \neg\alpha \vee A)$ is guarded, since a is a guard, and A is not affected. Any quantification in which the replaced formula $\forall\overline{x}\,(\neg a \vee A)$ occurs, remains guarded after replacement by $\alpha(\overline{y})$, because no new free variables are introduced.

In the result of $\mathrm{Struct}_\forall$ there are no nested, universal quantifications. Because of this, every existential quantifier is in the scope of at most one universal quantification, which is guarded. The result of the Skolemization is a formula in which all universal quantifiers are guarded, and all functional terms are Skolem terms. They are either constants or contain all variables of the guarded quantification in which they occur.
Clearly, at the end of this process the formulae $\forall \overline{x}\, (\neg a \vee A)$ can be factored into guarded clauses $\forall \overline{x}\, (\neg a \vee A_1), \ldots, \forall \overline{x}\, (\neg a \vee A_n)$. $\Box$

3.2. Termination

As announced in the previous section, the first step towards our decidability result for the guarded fragment will be to show that, with a suitable ordering, ordered resolution terminates for the guarded fragment.
We will now define the order on literals. Although we will be using completely standard ordered resolution, our order is non-standard.

Definition 3.6. *We define the following order $\sqsubset$ on literals.*

1 $A \sqsubset B$ *if* $\mathrm{Vardepth}(A) < \mathrm{Vardepth}(B)$, *or*
2 $A \sqsubset B$ *if* $\mathrm{Var}(A) \subset \mathrm{Var}(B)$.

Note that the inclusion in the second condition is strict. Strictly seen we cannot call relation $\sqsubset$ an order because it is not transitive. However, $\sqsubset$ is an order within guarded clauses, in particular it has the following property:

Lemma 3.7. *Every guarded clause c has a $\sqsubset$-maximal literal, and every maximal literal of c contains all variables of c.*

Proof. If c is ground, then every literal is maximal. If c is non-ground, and does not contain a non-ground functional term, then every guard is maximal, since it contains all variables of c and there are no deeper literals. If c is non-ground, and does contain non-ground, functional terms, then there are literals containing the deepest occurrence of a non-ground, functional term. These literals must be maximal, because they contain all variables of c.
If c is non-grond there is a literal containing all variables of c. Because of this every maximal literal must also contain all variables of c. $\Box$

The result that we aim to prove is that resolution and factoring, restricted by $\sqsubset$, can only derive a finite set of clauses from a guarded clause set, but first we prove that the property of being guarded is preserved.

Theorem 3.8. *1 If c_1 and c_2 are guarded clauses, and c is a $\sqsubset$-ordered resolvent of c_1 and c_2, then c is guarded.*
2 If c_1 is a guarded clause, and c is a factor of c_1, then c is guarded.

We show that derived clauses satisfy Definition 3.1. We first show Condition 1, then Condition 2.
Claim 1 Condition 1 is preserved by resolution and factoring.

PROOF. Let $c_1 = \{A_1\} \cup R_1$ and $c_2 = \{A_2\} \cup R_2$ resolve into $c = R_1\Theta \cup R_2\Theta$, so Θ is the mgu of A_1 and A_2. Because of the order $\sqsubset$, the literals A_1 and A_2 contain all variables of their respective clauses. This ensures that $A_1\Theta = A_2\Theta$ contains all variables of the resolvent c. Because both A_1 and A_2 are weakly covering, every non-ground functional term in $A_1\Theta$ contains all variables of $A_1\Theta$ and hence of c.
Let t be a non-ground functional term in c. There are two possibilities:

1 There is a non-ground functional term u in c_1 or c_2, such that $t = u\Theta$. W.l.o.g. assume that u occurs in c_1. Then u contains all variables of A_1. Because of this, $u\Theta$ contains all variables of $A_1\Theta$. Since $A_1\Theta$ contains all variables of c, the term $t = u\Theta$ contains all variables of c.
2 There is a variable V in c_1 or c_2, such that t is a subterm of $V\Theta$. Assume w.l.o.g that V occurs in c_1. Then V also occurs in A_1. Hence t, being a subterm of $V\Theta$, occurs in $A_1\Theta$. This makes that t contains all variables of c.

Next let $c = \{A_1\Theta\} \cup R\Theta$ be a factor of $c_1 = \{A_1, A_2\} \cup R$. Analogous to the situation with resolution, one of the literals A_1, A_2 contains all variables of c_1. Assume it is A_1. The situation is the same as with resolution: $A_1\Theta = A_2\Theta$ contains all variables of c, every non-ground functional term in $A_1\Theta$ contains all variables of c, etc. However, case 2 is not possible here (there exists a variable V in c_1, such that t occurs in $V\Theta$) because the variable V would occur in A_1. This contradicts $\text{Vardepth}(A_1\Theta) \leq \text{Vardepth}(A)$. $\square$

Claim 2 Condition 2 is preserved.

PROOF. First we consider resolution. If both c_1, c_2 are ground, then c is also ground, and hence satisfies Condition 2. If one of c_1, c_2 is ground, then assume it is c_1. Because A_2 contains all variables of c_2, and $A_2\Theta$ is ground, the resolvent c is also ground in this case. Now if both c_1 and c_2 are not ground, then let $\neg G_1$, $\neg G_2$ be guards of c_1, c_2. In one of c_1, c_2, the guard is not resolved upon, because guards are negative. We can assume that $A_1 \neq G_1$.

1 If Θ does not assign a non-ground, functional term to any variable in A_1, then $\neg G_1\Theta$ is a guard of c, because $\neg G_1\Theta$ does not contain any non-ground, functional terms, and due to the fact that G_1 contains the same variables of A_1, the result $\neg G_1\Theta$ contains all variables of $A_1\Theta$, which contains all variables of c, by the proof of the first claim.
2 Otherwise, Θ assigns a non-ground, functional term to a variable in A_1. This is caused by the fact that A_2 contains a non-ground, functional term, which implies that $A_2 \neq G_2$. Then Θ does not assign a non-ground, functional term to any variable in A_2. This makes that $\neg G_2\Theta$ can act as guard of c, by the same argument as before.

The situation with factoring is the same. One of A_1, A_2 contains all variables of c_1. Because of this, the mgu Θ cannot assign a non-ground, functional term to a variable in c_1. This implies that every guard of c_1 is still a guard of c. $\square$

In fact, one can prove that factoring without $\sqsubset$ also preserves the guarded fragment. However, in case of resolution one really needs the $\sqsubset$-order.

LEMMA 3.9. *Let C be a finite set of guarded clauses. Let $v = \mathrm{Vardepth}(C)$. Let k be the maximal* $\mathrm{Varnr}(c)$, *for $c \in C$. Then for every $\sqsubset$-derivable clause c the following holds:*

1 $\mathrm{Varnr}(c) \leq k$.
2 $\mathrm{Vardepth}(c) \leq v$.

PROOF. We first prove the first fact. Let c be the resolvent of c_1 and c_2. If either of c_1 or c_2 is ground, then c is ground by itself. If both c_1 and c_2 are non-ground, then c contains a guard $\neg A$, which is an instance of a guard of either c_1 or c_2. We can assume that $\mathrm{Varnr}(c_1), \mathrm{Varnr}(c_2) \leq k$. Since every variable of c occurs in $\neg A$, and $\mathrm{Varnr}(\neg A) \leq k$, we immediately obtain $\mathrm{Varnr}(c) \leq k$. The case where c is obtained by factoring is immediate. In order to prove the second fact, let c be the resolvent of $c_1 = \{A_1\} \cup R_1$ and $c_2 = \{A_2\} \cup R_2$. By induction there is no literal with Vardepth $> v$ in c_1 or c_2. Assume that $\mathrm{Vardepth}(A_1) \geq \mathrm{Vardepth}(A_2)$. Let Θ be the unifier used. By Lemma 2.10 we have $\mathrm{Vardepth}(A_1\Theta) \leq \mathrm{Vardepth}(A_1)$. By Lemma 2.13, we have $\mathrm{Vardepth}(R_i\Theta) \leq \mathrm{Vardepth}(A_i)$. It follows that $\mathrm{Vardepth}(R_1\Theta \cup R_2\Theta) \leq v$. The case where c is obtained by factoring is analogous. □

We would have the proof complete if we would have $\mathrm{Depth}(\overline{C}) \leq \mathrm{Depth}(C)$. Unfortunately this is not the case, but it is possible to prove that no new ground terms are introduced at positions that are deeper than $\mathrm{Vardepth}(C)$.

LEMMA 3.10. *1 Let c be a $\sqsubset$-ordered resolvent of clauses c_1 and c_2. Let v be the greater of* $\mathrm{Vardepth}(c_1)$ *and* $\mathrm{Vardepth}(c_2)$. *Every ground term t that occurs at a depth greater than or equal to v, occurs either in c_1 or in c_2.*
2 Let c be a factor of clause c_1. Let $v =$ $\mathrm{Vardepth}(c_1)$. *Every ground term occurring in c at a depth greater than or equal to v, occurs in c_1.*

PROOF. 1 Write $c_1 = \{A_1\} \cup R_1$, and $c_2 = \{\neg A_2\} \cup R_2$. Let Θ be the mgu of A_1 and A_2. We can assume, without loss of generality, that t occurs in $R_1\Theta$. There are two possibilities:

(a) There is a variable V in R_1, such that t is a subterm of $V\Theta$, or $t = V\Theta$. When this is the case, V occurs in A_1, at least as deep as in R_1. This ensures that t occurs in A_1, at a depth greater than or equal to v. Hence we can apply Lemma 2.11, and it follows that t occurs in A_1 or A_2.
(b) There is a term u in R_1, such that $t = u\Theta$, and u is not a variable. If u is ground, then we are done. If u is non-ground, then u contains variables at depth greater than v. This implies that $\mathrm{Vardepth}(c_1) > v$, so this cannot occur.

2 The case where c is obtained by factoring is analogeous.
□

From Lemma 3.10 an upperbound on the depth of the derivable clauses can be easily obtained. Let C be the initial clause set. Let $v = \mathrm{Vardepth}(C)$ and let $d = \mathrm{Depth}(C)$. Let c be some derivable clause. Since every term occurring at depth $\geq v$ occurs in C, it has a depth $\leq d$. Hence $\mathrm{Depth}(c) \leq v + d$.

LEMMA 3.11. *Let C be a finite set of guarded clauses. Let $\overline{C}$ be its closure under $\sqsubset$-ordered resolution, and (unrestricted) factoring. Then $\overline{C}$ has finite size.*

PROOF. For each derivable clause, both the depth and the number of variables are bounded. □

We will derive the exact complexity of the decision procedure in Section 3.4

3.3. Completeness

The final step in our proof of the decidability of the guarded fragment by means of resolution consists of proving completeness of our ordered resolution method. The $\sqsubset$-order is non-liftable. Both cases in Definition 3.6 cause non-liftability:

1 $p(s(0), X) \sqsubset p(0, s(X))$ and $p(X, 0) \sqsubset p(s(X), s(0))$. The substitution $\{X := 0\}$ results in a conflict.
2 Also $\neg p(X, X) \sqsubset \neg q(X, Y)$ and $\neg q(X, X) \sqsubset \neg p(X, Y)$. The substitution $\{X := Y\}$ results in a conflict.

Because of this we cannot refer to the standard result on the completeness of liftable orders. Also the completeness results in (de Nivelle, 1994) do not apply because there one of the following two conditions should have been met:

1 The order needs to satisfy the property $A\Theta \sqsubset A$, for non-renaming substitutions Θ. Our order puts $A(X) \sqsubset A(s(X))$, but $A(s(X))$ is an instance of $A(X)$.
2 The literals in the clauses must have the same set of variables. The guarded clause $\{\neg a(X, Y), b(X)\}$ violates this condition.

Fortunately however, although guarded clauses do not satisfy Condition 2, it turns out that the proof method that was used for Condition 2, can be applied to guarded clauses. The proof is based on the resolution game. We need some technical preparation.

DEFINITION 3.12. *A* representation-indexed clause *is a clause of the form* $c = \{a_1{:}A_1, \ldots, a_p{:}A_p\}$ *for which there exists a substitution* Θ, *such that* $A_i\Theta = a_i$, *for all* i. *If for each variable* V *that does not occur in an* A_i, *it is the case that* $V\Theta = V$, *then we call* Θ the substitution of c. *A literal order* $\sqsubset$ *can be extended to indexed literals as follows:*

$$a{:}A \sqsubset b{:}B \textit{ iff } A \sqsubset B.$$

Using this we extend ordered resolution and ordered factoring to representation-indexed clauses as follows:

Resolution: *From* $\{a{:}A_1\} \cup r_1{:}R_1$ *and* $\{\neg a{:}A_2\} \cup r_2{:}R_2$ *derive* $r_1{:}R_1\Theta \cup r_2{:}R_2\Theta$.
Factoring: *From* $\{a{:}A_1, a{:}A_2\} \cup r{:}R$ *derive* $\{a{:}A_1\Theta\} \cup r{:}R\Theta$.

In both cases Θ *is the mgu. The literals resolved upon, and one of the literals factored upon must be maximal. Observe that the mgu always exists.*

LEMMA 3.13. *Let* C_1 *be a set of representation-indexed clauses, that has a resolution refutation, using some order* $\sqsubset$ *. Let* C_2 *be obtained from* C_1 *by replacing each representation-indexed clause* $\{a_1{:}A_1, \ldots, a_p{:}A_p\}$ *by* $\{A_1, \ldots, A_p\}$. *Then* C_2 *has a resolution refutation using* $\sqsubset$.

Proof. One can delete the ground instance from every derivable representation-indexed clause, and show that it is still derivable. □

We will construct a resolution game from a set of representation-indexed clauses. In order to do this we define an operator [ ] from representation-indexed clauses to indexed clauses of the type used in the resolution game. Before we can define [ ], we need the following:

Definition 3.14. *We assume that there is a fixed enumeration of the set of variables* $\{X_0, X_1, X_2, \ldots\}$. *A literal* A *is* normal *if the variable* X_{i+1} *occurs only after an occurrence of the variable* X_i. *(When the literal is written in the standard notation). Every literal* A *can be renamed into exactly one normal literal, which we call the* normalization *of* A. *We write* $\overline{A}$ *for the normalization of* A.

The literal $p(X_0, X_1, X_2)$ is normal, but its renamings $p(X_1, X_0, X_2)$ and $p(X_1, X_2, X_3)$ are not normal. If two literals are renamings of each other, they have the same normalization.

Lemma 3.15. *Let* $\sqsubset$ *be the order of Definition 3.6. If* $A \sqsubset B$ *then* $\overline{A} \sqsubset \overline{B}$.

Definition 3.16. *Let* $\Theta = \{V_1 := t_1, \ldots, V_n := t_n\}$ *be a substitution. The complexity of* Θ, *written as* $\#\Theta$ *equals* $\#t_1 + \cdots + \#t_n$.

Definition 3.17. *We define the following operator* [ ] *on representation-indexed clauses. Let* $\{a_1 : A_1, \ldots, a_p : A_p\}$ *be a representation-indexed clause. Let* Θ *be its substitution. Let* $k = \#\Theta$. *Then*

$$[\{a_1 : A_1, \ldots, a_p : A_p\}]$$

equals the indexed clause

$$\{a_1 : (k, \overline{A}_1), \ldots, a_p : (k, \overline{A}_p)\}.$$

The $\overline{A}_1, \ldots, \overline{A}_p$ *are the normalizations of the* $A_1, \ldots, A_p$.

Lemma 3.18. *Let* $c_1 = \{a_1 : A_1, \ldots, a_p : A_p\}$ *be a representation-indexed clause. Let* $c_2 = \{a_1 : A_1\Sigma, \ldots, a_p : A_p\Sigma\}$ *be an instance obtained with substitution* Σ, *such that there exists a substitution* Ξ, *for which* $a_i = A_i\Sigma\Xi$. *Let*

$$[c_1] = \{a_1 : (k_1, \overline{A_1}), \ldots, a_p : (k_1, \overline{A_p})\},$$

$$[c_2] = \{a_1 : (k_2, \overline{A_1\Sigma}), \ldots, a_p : (k_2, \overline{A_p\Sigma})\}.$$

Then either for all i, $\overline{A_i\Sigma} = \overline{A_i}$, *or* $k_2 < k_1$.

We are now ready for the completeness proof.

Theorem 3.19. *Ordered resolution, using* $\sqsubset$ *as defined in Definition 3.6, is complete for guarded clause sets.*

Proof. Let C be an unsatisfiable guarded clause set. Let $\overline{C}$ be the set of clauses that can be obtained from C using $\sqsubset$-ordered resolution, and $\sqsubset$-ordered factoring. We show that $\overline{C}$ must contain the empty clause. Write $C = \{c_1, \ldots, c_n\}$. Let

$$\begin{array}{c}\Theta_{1,1}, \ldots, \Theta_{1,l_1}, \\ \vdots \\ \Theta_{n,1}, \ldots, \Theta_{n,l_n}\end{array}$$

be a list of substitutions such that the set of clauses

$$\{c_1\Theta_{1,1}, \ldots, c_1\Theta_{1,l_1}, \ldots, c_n\Theta_{n,1}, \ldots, c_n\Theta_{n,l_n}\}$$

is propositionally unsatisfiable. Such a set exists because of Herbrand's theorem. First we construct a set C_{hb} of representation-indexed clauses, using the Herbrand set. For each $c_i = \{A_1, \ldots, A_p\}$ and substitution $\Theta_{i,j}$, the set C_{hb} contains the clause

$$\{A_1\Theta_{i,j}{:}\, A_1, \ldots, A_p\Theta_{i,j}{:}\, A_p\}.$$

Next we write $\overline{C}_{hb}$ for the closure of C_{hb} under $\sqsubset$-ordered resolution for representation-indexed clauses. It is clear from Lemma 3.13 that if we can prove that $\overline{C}_{hb}$ contains the empty clause, then $\overline{C}$ contains the empty clause. In order to prove that $\overline{C}_{hb}$ does indeed contain the empty clause, we define the following resolution game $\mathcal{G} = (P, \mathcal{A}, \prec)$, and initial clause set $C_{\mathcal{G}}$:

1 The set P of propositional symbols equals the set of atoms that occur as a in the elements $a{:}\, A$ of C_{hb}.
2 The set $\mathcal{A}$ of attributes is constructed as follows: Let m be the maximal $\#\Theta_{i,j}$. Let L be the set of literals B for which there is an indexed literal $a{:}\, A$ in one of the $\overline{C}_{hb}$, such that B is an instance of A, and a is an instance of B. Then $\mathcal{A}$ consists of the pairs (i, C), for which $0 \leq i \leq m$, and C is the normalization of a literal in L. Observe that the set of attributes is finite.
3 The order $\prec$ is defined from: $a_1{:}\,(i_1, C_1) \prec a_2{:}\,(i_2, C_2)$ if
   (a) $i_1 < i_2$, or
   (b) ($i_1 = i_2$ and $C_1 \sqsubset C_2$).
4 The initial clause set $C_{\mathcal{G}}$ equals $\{\ [c] \mid c \in C_{hb}\}$.

This completes the definition of the resolution game. We will complete the proof by showing that the set

$$[\overline{C}_{hb}] = \{[c] \mid c \text{ is derivable from } C_{hb}\}$$

is a saturation of $(P, \mathcal{A}, \prec)$. Then it follows from Theorem 2.18, that $[\overline{C}_{hb}]$ contains the empty clause. From this it follows immediately that $\overline{C}_{hb}$ contains the empty clause.
It remains to show that $[\overline{C}_{hb}]$ is a saturation of $(P, \mathcal{A}, \prec)$. In order to do this we must show that $[\overline{C}_{hb}]$ contains a reduction of every factor/resolvent that is derivable from $[\overline{C}_{hb}]$.

1 Let c_1 and c_2 be clauses in $[\overline{C}_{hb}]$ with a resolvent c. There must exist clauses $d_1, d_2 \in C_{hb}$, such that $c_1 = [d_1]$, and $c_2 = [d_2]$. Write

$$d_1 = \{a{:}\, A_1\} \cup r_1{:}\, R_1 \text{ and } d_2 = \{\neg a{:}\, A_2\} \cup r_2{:}\, R_2.$$

Then we can write

$$c_1 = \{a{:}(k_1, \overline{A_1})\} \cup r_1{:}(k_1, R_1) \text{ and } c_2 = \{\neg a{:}(k_2, \overline{A_2})\} \cup r_2{:}(k_2, R_2).$$

We use the notation $r_i{:}(k_i, R_i)$ for the side (indexed) literals. They have the form

$$[r_{i,1}{:}(k_{i,1}, R_{i,1}), \ldots, r_{i_{l_i}}{:}(k_{i,l_i}, R_{i,l_i})].$$

Using Lemma 3.15, we obtain that the indexed literals $a{:}A_1$ and $\neg a{:}A_2$ are maximal in their respective clauses. Hence a resolvent

$$d = r_1{:}R_1\Theta \cup r_2{:}R_2\Theta$$

is possible, where Θ is the mgu. We will show that $[d]$ is a reduction of c. Let Σ be the substitution of the representation-indexed clause d. Let Σ_1 be the substitution of the representation-indexed clause

$$d_1\Theta = \{a{:}A_1\Theta\} \cup r_1{:}R_1\Theta.$$

Analogously let Σ_2 be the substitution of the representation-indexed clause

$$d_2\Theta = \{\neg\ a{:}A_2\Theta\} \cup r_2{:}R_2\Theta.$$

By putting $l = \ \#\ \Sigma$, we can write

$$[d] = r_1{:}(l, \overline{R_1\Theta}) \cup r_2{:}(l, \overline{R_2\Theta}).$$

Write $l_1 = \#\Sigma_1,\ l_2 = \#\Sigma_2$. Then

(a) $r_1{:}(l_1, \overline{R_1\Theta})$ is a reduction of $r_1{:}(k_1, R_1)$, using Lemma 3.18.
(b) $r_2{:}(l_2, \overline{R_2\Theta})$ is a reduction of $r_2{:}(k_2, R_2)$, using Lemma 3.18.
(c) $l \le l_1$ and $l \le l_2$.

Putting this together we obtain that $[d]$ is a reduction of c.

2 Finally, in the second case, where a clause c_1 has a factor c in $[C_{hb}]$ we can directly apply Lemma 3.18.

□

The order $\sqsubset$ as we have defined it in Definition 3.6 is very basic, and it could be strengthened further to improve the efficiency, for example with an order on the predicate symbols.

THEOREM 3.20. *Resolution + factoring, using $\sqsubset$, together with the normal form transformation of Theorem 3.5, is a decision procedure for the guarded fragment.*

PROOF. Follows from Theorem 3.5, Lemma 3.11 and Theorem 3.19. □

3.4. Complexity

The complexity of our decision procedure is double exponentional. Grädel has shown in (Grädel, 1997) that the decision problem for the guarded fragment is 2EXPTIME-complete, so our procedure is theoretically optimal. First we give a general bound on the time needed to compute a saturation.

LEMMA 3.21. *Let C be some clause set, let $\overline{C}$ be its closure under resolution and factoring. Let S be some clause set, such that $\overline{C} \subseteq S$. Let s be the maximal size of a clause in S. Let c be the cardinality of S. Then $\overline{C}$ can be computed in time $c(cs)^2$ and space cs.*

PROOF. The space complexity is dominated by the space that is needed to store $\overline{C}$. The space needed to store S equals at most cs, and this is also an upperbound for the size of $\overline{C}$.
In order to obtain a saturation, the algorithm has to systematically inspect all pairs of clauses and to see if a resolvent or factor is possible. The cost is $cs.cs + cs$, which is dominated by $(cs)^2$. The algorithm halts when no more clauses can be added. This is the case after at most c iterations. □

THEOREM 3.22. *Let $\mathcal{S}$ be some signature. Let C be a set of guarded clauses over $\mathcal{S}$, possibly using variables. Let v be the maximal vardepth of a clause in C, and let $\mathcal{G}$ be the set of ground terms that occur in C. Let a be the maximal arity of a predicate/function symbol in $\mathcal{S}$. Let n be the maximum of* **(1)** *the total number of function symbols + the maximal arity of a guard + the size of $\mathcal{G}$, and* **(2)** *the total number of 0-arity predicate symbols. Then a saturation of C has at most size*

$$2n^{(a^v)},$$

and can be obtained in time

$$2^{3(2n^{(a^v)})}.$$

PROOF. Using Lemma 3.10, and Lemma 3.9, we know that at positions at depth v or deeper, there are only ground terms from $\mathcal{G}$. Hence we can treat the literals in the saturation of C as if they have a depth of $v + 1$, and view the $\mathcal{G}$ as additional constants. Define the following numbers:

a_1	be the maximal arity of a predicate symbol,
a_2	be the maximal arity of a function symbol,
n_1	be the total number of function symbols + the total nunber of constant symbols + the maximal arity of a guard.
n_2	be the total number of predicate symbols

We begin by giving an estimation of the number of positions $P(d)$ in a term, dependent on its depth d. The second column in the table gives $P(i)$ defined in terms of $P(i-1)$. The third column gives explicit forms for $P(i)$.

d		
1	1	$1,$
2	$1 + a_1P(1)$	$1 + a_1,$
3	$1 + a_1P(2)$	$1 + a_1 + a_1^2,$
4	$1 + a_1P(3)$	$1 + a_1 + a_1^2 + a_1^3.$

So we get

$$P(d) = \sum_{i=0}^{d-1} a_1^i = \frac{a_1^d - 1}{a_1 - 1} \approx O(a_1^{d-1}), \text{ when } a_1 > 1.$$

The number of terms of depth d then can be estimated by

$$(n_1)^{(a_1^{d-1})}.$$

We could write $n_1 + 1$ instead of n_1 because positions can be empty, when the term does not use the full possible length, but in that case there is an operator that does not use the full a_1, which compensates for this.
A literal of depth d consists of a possible negation sign, followed by one predicate symbol, followed by at most a_2 terms with depth $d-1$. The number of possible literals can be estimated by

$$2n_2(n_1^{(a_1^{d-2})})^{a_2}.$$

By remembering that $n = \mathrm{Max}(n_1, n_2)$, $a = \mathrm{Max}(a_1, a_2)$, and putting $d = v + 1$, we can estimate the number of possible literals as

$$2n^{(a^v)}.$$

Then the set of possible clauses has at most size

$$2^{(2n^{(a^v)})}.$$

Applying Lemma 3.21, we obtain the given space and time complexity. □

4. The Loosely Guarded Fragment

In this section we show that the loosely guarded fragment can also be decided by resolution. The loosely guarded fragment is a generalization of the guarded fragment, which has been introduced in (van Benthem, 1997). The guard no longer needs to be a single literal as in the guarded fragment, but may consist of a group of literals satisfying certain conditions. One of the main motivations behind the loosely guarded fragment is the following. Recall that one of the motivations behind the original guarded fragment was the search for general fragments of first-order logic that could explain the good behavior of modal and modal-like logics. An important and well-behaved *temporal* logic that escapes the guarded fragment is temporal logic with the Since and Until operators. Recall that the semantics of P Until Q is given by the following definition:

$$\exists y\, (Rxy \wedge Qy \wedge \forall z\, (Rxz \wedge Rzy \rightarrow Pz)).$$

Clearly, this is not a guarded formula, but it does enjoy a special property: the variable z occurs together with each of the other variables x and y in at least one atom in the 'loose guard.' This special feature motivates the following definition.

DEFINITION 4.1. *The* loosely guarded fragment *(LGF) is recursively defined as the following subset of first-order logic without equality and function symbols.*

1. $\top$ *and* $\bot$ *are in* LGF.
2. *If* A *is an atom, then* $A \in$ LGF.

3 If $A \in$ LGF, *then* $\neg A \in$ LGF.
4 If $A, B \in$ LGF, *then* $A \vee B,\ A \wedge B,\ A \rightarrow B,\ A \leftrightarrow B \in$ LGF.
5 (a) Let $A \in$ LGF,
(b) let $a_1, \ldots, a_n$ *be a group of atomic formulae,*
(c) let $\overline{x}$ *be a sequence of variables,*
such that for every variable in $\overline{x}$, *and for every free variable of* $a_1 \wedge \cdots \wedge a_n \rightarrow A$, *there is an* a_i *containing them both. Then* $\forall \overline{x}(a_1 \wedge \cdots \wedge a_n \rightarrow A) \in$ LGF, *and* $\exists \overline{x}(a_1 \wedge \cdots \wedge a_n \wedge A) \in$ LGF. *We also allow* $\forall \overline{x}(\neg a_1 \vee \cdots \vee \neg a_n \vee A) \in$ LGF.

The definition of LGF can be weakened in the same way as GF, if one considers the satisfiability problem. The guard condition is only necessary for positively occurring $\forall$-quantifiers, and for negatively occurring $\exists$-quantifiers. The guarded fragment is included in the loosely guarded fragment.

Example 4.2. The transitivity axiom

$$\forall xyz(R(x,y) \wedge R(y,z) \rightarrow R(x,z))$$

is not not loosely guarded, because an atom containing both x and z is missing. The following formula, translating P Since Q, is loosely guarded:

$$\exists y(Ryx \wedge Qy \wedge \forall z(Ryz \wedge Rzx \rightarrow Pz)).$$

4.1. Translation to CNF

The strategy that we will use for LGF is based on the strategy for GF. The transformation to CNF will be almost the same, with an obvious adaption in $\mathrm{Struct}_\forall$ to handle loose guards. The resolution strategy will be more involved, as will discuss in the next section. We now introduce LGF for clauses, and the transformation.

Definition 4.3. *A clause set is called* loosely guarded *if its clauses are loosely guarded. A clause c is* loosely guarded *if it satisfies the following condition:*

1 Every non-ground, functional term in c contains all variables of c.
2 If c is non-ground, then there is a set of negative literals $\neg A_1, \ldots, \neg A_p \in c$ *not containing non-ground, functional terms, such that every pair* X, Y *of variables of c occurs together in at least one of the* $\neg A_i$.

The conjunction of the atoms A_i in Item 2 is the loose guard. A clause may have more than one loose guard.

Theorem 4.4. *Using the following transformation, loosely guarded formulae can be translated into loosely guarded clause sets:*

1 $F' = \mathrm{NNF}(F)$.
2 $F'' = (\mathrm{Struct}_\forall)(F')$.
3 $C = (\mathrm{Sk}; \mathrm{Cls})(F'')$.

(Here, $\mathrm{Struct}_\forall$ *has been modified in the obvious way)*

Proof. The proof is analogous to the proof of Theorem 3.5. However, there is one interesting aspect concerning $\mathrm{Struct}_\forall$. Transformation $\mathrm{Struct}_\forall$ replaces universally quantified subformulae $\forall\overline{x}(\neg a_1 \vee \cdots \vee \neg a_n \vee A)$ with free variables $\overline{y}$ by a fresh atom $\alpha(\overline{y})$ and introduces a definition

$$\forall\overline{xy}(\neg a_1 \vee \cdots \vee \neg a_n \vee \neg\alpha(\overline{y}) \vee A).$$

Then the disjunction

$$\neg a_1 \vee \cdots \vee \neg a_n \vee \neg\alpha(\overline{y})$$

is a loose guard. To see this, let v_1, v_2 be a pair of variables occurring in $\overline{xy}$. If either v_1 or v_2 is among the $\overline{x}$, then v_1 and v_2 occur together in one of the $\neg a_i$, because the original quantification was loosely guarded. If both v_1 and v_2 are not among the $\overline{x}$, then they are both among the $\overline{y}$, and then they occur together in $\neg\alpha(\overline{y})$. $\Box$

4.2. Termination

The ordering strategy for loosely guarded clause sets is more complicated than the decision procedure for guarded clause sets. This is caused by problems that occur when we have to select the literals of the loose guard. The completeness proof of Theorem 3.19 hinges on the fact that it is always possible to select a literal containing all variables of the clause. This is not possible with loosely guarded clauses, because such a literal may not exist, as for example in clause c_0 below. The obvious approach would be to use the closest possible approximation of the strategy for the guarded fragment. When there are literals with non-ground functional terms, prefer the literals with maximal Vardepth. When there are no literals with non-ground functional terms, select the complete loose guard and resolve it away using hyperresolution (see Definition 2.4). Unfortunately at this point growth of Vardepth is possible, as can be seen from the following example:

Example 4.5. The following clause is loosely guarded:

$$c_0 = \{\neg a_1(X,Y), \neg a_2(Y,Z), \neg a_3(Z,X),\ b_1(X,Y), b_2(Y,Z), b_3(Z,X)\}.$$

There are no non-ground functional terms, so the clause is a candidate for hyperresolution. It is possible to construct a hyperresolvent with the following clauses

$$\begin{array}{rcl} c_1 & = & \{\neg p_1(A), a_1(s(A), s(A))\}, \\ c_2 & = & \{\neg p_2(B), a_2(B, t(B))\}, \\ c_3 & = & \{\neg p_3(C), a_3(t(C), C)\}, \end{array}$$

using the substitution

$$\Theta = \{X, Y, B, C := s(A), Z := t(s(A))\}.$$

The result equals

$$\begin{array}{l} \{\neg p_1(A), \neg p_2(s(A)), \neg p_3(s(A)), \\ \qquad b_1(s(A), s(A)),\ b_2(s(A), t(s(A))),\ b_3(t(s(A)), s(A))\}, \end{array}$$

which has a Vardepth of 2, which is too deep.

Here is an explanation for the problem of Example 4.5. Clause c_0 can hyperresolve with clauses c_2 and c_3 using substitution

$$\Theta = \{Y, B, X := C, Z := t(C)\}.$$

The result equals:

$$c_{part} = \{\neg a_1(C, C), \neg p_2(C), \neg p_3(C), b_1(C, C), b_2(C, t(C)), b_3(t(C), C)\}.$$

This clause is loosely guarded, and it is not too deep. To obtain the final hyperresolvent one needs to resolve upon the literal $\neg a_1(C, C)$. However, $a_1(C, C)$ is not the deepest term in the clause, and when $a_1(C, C)$ is unified with $a_1(s(A), s(A))$ the literal $b_2(c, t(C))$ grows into a Vardepth of 2. This means that our refinement should allow the construction of c_{part}, but that it should block resolving c_{part} with c_1.
Instead of allowing the construction of full hyperresolvents, we allow the construction of partial hyperresolvents that are not too deep. We will prove that whenever a hyperresolvent can be found using the loose guard, there exists a partial hyperresolvent which does not grow in Vardepth and which is loosely guarded. In order to do this, we need to go into details of how the mgu is constructed. For this purpose we repeat the following algorithm for the construction of most general unifiers. It comes from (Fermüller *et al.*, 1993).

DEFINITION 4.6. *The following algorithm decides whether or not two literals A and B have a unifier. It constructs a most general unifier if there exists a unifier.*

First, we define the notion of a minimal difference *of two literals. Let A and B be two literals, such that $A \neq B$. A minimal difference is a pair (A', B') that is the result of the following decomposition:*

1 *Put $A' := A$, and $B' := B$.*
2 *As long as A' has the form $p(t_1, \ldots, t_n)$ and B' has the form $p(u_1, \ldots, u_n)$, replace A' by t_i and B' by u_i, for an i, such that $t_i \neq u_i$.*

Using this, the algorithm for computing mgu's is defined as follows. Let A and B be the terms to be unified. Put $\Theta := \{\ \}$, the identity substitution.

1 *If $A = B$, then Θ equals the most general unifier.*
2 *As long as $A \neq B$, let (A', B') be a minimal difference. Then*
   - *(a) If (A', B') has the form $(p(t_1, \ldots, t_n),\ q(u_1, \ldots, u_m))$, with $p \neq q$, or $n \neq m$, then report failure.*
   - *(b) If (A', B') has the form (V, t), where V is a variable, $V \neq t$, but V occurs in t, then report failure.*
   - *(c) If (A', B') has the form (t, V), where V is a variable, $V \neq t$, but V occurs in t, then report failure.*
   - *(d) If (A', B') has the form (V, t) where V is a variable, and V does not occur in t, then put $A := A\{V := t\}$, $B := B\{V := t\}$, $\Theta := \Theta \cdot \{V := t\}$.*
   - *(e) If (A', B') has the form (t, V), where V is a variable, and V does not occur in t, then put $A := A\{V := t\}$, $B := B\{V := t\}$, $\Theta := \Theta \cdot \{V := t\}$.*

The procedure of Definition 4.6 is complete and sound. Up to renaming, the result does not depend on the choice of the minimal difference. See (Fermüller *et al.*, 1993) for details.

THEOREM 4.7. *Assume that the literals $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$ and the substitution Θ satisfy the following conditions:*

1. *All A_i have no non-ground, functional terms.*
2. *For all $X, Y \in \mathrm{Var}(A_1, \ldots, A_n)$ there is an A_i such that $X, Y \in \mathrm{Var}(A_i)$.*
3. *All B_j are weakly covering and have a non-ground, functional term.*
4. *If $i \neq j$, then B_i and B_j have no overlapping variables.*
5. *There are no overlapping variables between the A_i and the B_j.*
6. *Θ is the mgu of $(A_1, B_1), \ldots, (A_n, B_n)$.*

Then it is possible to find a permutation $(\pi_1, \ldots, \pi_n)$ with the following properties: Write

$$(A'_1, \ldots, A'_n) = (A_{\pi_1}, \ldots, A_{\pi_n})$$

and

$$(B'_1, \ldots, B'_n) = (B_{\pi_1}, \ldots, B_{\pi_n}).$$

There exists an $m \leq n$, such that, when Θ' is the mgu of $(A'_1, B'_1), \ldots, (A'_m, B'_m)$, then

1. $\mathrm{Varnr}(B'_1\Theta') \leq \mathrm{Varnr}(B'_1)$, *and* $\mathrm{Vardepth}(B'_1\Theta') \leq \mathrm{Vardepth}(B'_1)$.
2. *For all i, with $1 \leq i \leq m$,*
$$\mathrm{Var}(B'_i\Theta') \subseteq \mathrm{Var}(B'_1\Theta'), \text{ and } \mathrm{Vardepth}(B'_i\Theta') \leq \mathrm{Vardepth}(B'_1\Theta').$$
3. *For all i, with $1 \leq i \leq m$,*
$$\mathrm{Var}(A'_i\Theta') = \mathrm{Var}(B'_i\Theta'), \text{ and } \mathrm{Vardepth}(A'_i\Theta') = \mathrm{Vardepth}(B'_i\Theta').$$
4. *For all i, with $1 \leq i \leq m$, both $A'_i\Theta'$ and $B'_i\Theta'$ are weakly covering.*

As a consequence, B'_1 limits the complexity of the result.

PROOF. Item 3 follows immediately from the fact that Θ' is a unifier. Before we can establish items 1 and 2 we need the following notion. When a variable V occurs as $A_i(\ldots, V, \ldots)$, and a term t as $B_i(\ldots, t, \ldots)$, we say that V is *paired* to t.
If all $A_i\Theta$ are ground, then the theorem follows trivially. Otherwise, define the following order $\sqsubset$ on variables V that occur in the formulae $A_1, \ldots, A_n$ and for which $V\Theta$ is not ground:

$X \sqsubset Y$ if X and Y occur together in an A_i, as $A_i(\ldots, X, \ldots, Y, \ldots)$, and in the corresponding B_i there is $B_i(\ldots, T, \ldots, U, \ldots)$, with $\mathrm{Vardepth}(T) < \mathrm{Vardepth}(U)$.

Then the following property holds:

MAXVAR There exists a $\sqsubset$-maximal variable in $(A_1, \ldots, A_n)$

To see that **MAXVAR** holds, argue as follows. If there does not exist a maximal variable this is caused by the fact that there is a cycle as follows:

$$V_0 \sqsubset V_1 \sqsubset \cdots \sqsubset V_p \sqsubset V_0.$$

We show that in this case there does not exist a unifier. The cycle is caused by literals of the form:

$$A_0(V_0, V_1), A_1(V_1, V_2), A_2(V_2, V_3), \dots, A_p(V_p, V_0),$$

and

$$B_0(t_0, u_0), B_1(t_1, u_1), B_2(t_2, u_2), \dots, B_p(t_p, u_p),$$

with $\text{Vardepth}(t_i) < \text{Vardepth}(u_i)$. Because the t_i and u_i are weakly covering, $\text{Vardepth}(t_i\Theta) < \text{Vardepth}(u_i\Theta)$, ($t_i\Theta$ and $u_i\Theta$ need not be weakly covering, but that is not important) Because $u_i\Theta = V_{i+1}\Theta$, for $i < p$, and $u_p\Theta = t_0\Theta$ it follows that

$$\text{Vardepth}(t_0\Theta) < \text{Vardepth}(t_1\Theta) < \dots < \text{Vardepth}(t_p\Theta) < \text{Vardepth}(t_0\Theta),$$

which is impossible. This shows that **MAXVAR** holds.

We can now construct the permutation $(\pi_1, \dots, \pi_n)$. Let Z be a maximal variable under the $\sqsubset$-order. Define $(\pi_1, \dots, \pi_n)$ as the following permutation:

1 Permute the (A_i, B_i) where A_i contains Z before the (A_j, B_j), where A_j does not contain Z.
2 After that, sort the (A_i, B_i) by $\text{Vardepth}(B_i)$, putting the B_i with the largest Vardepth first.

Let m be the index of the last A_i that contains Z. Then the pairs (A'_i, B'_i) have the following property, for $1 \leq i \leq m$,

MAXVARDEPTH If Z is matched to a term t of B'_i in one of the (A'_i, B'_i), then $\text{Vardepth}(t) = \text{Vardepth}(B'_i)$.

Suppose for the sake of contradiction that there is a term u in B'_i, for which $\text{Vardepth}(u) > \text{Vardepth}(t)$. There are three possibilities:

1 u is paired to Z. In that case t and u have to be unified by Θ, which is impossible because $\text{Vardepth}(t) = \text{Vardepth}(u)$ and because of the fact that t and u are weakly covering.
2 u is paired to another variable, which contradicts the $\sqsubset$-maximality of Z, or
3 u is paired to a ground term. This would make $u\Theta$ ground. Since $\text{Vardepth}(u) > 0$, it follows that u contains all variables of B'_i. But then $B'_i\Theta$ is ground, and this contradicts the fact that $Z\Theta$ is non-ground.

Let Θ' be the mgu of the pairs

$$(A'_1, B'_1), \dots, (A'_m, B'_m).$$

We have to show that the permutation and Θ' have the desired properties 1 and 2. Write $\Theta' = \Sigma_1 \cdot \Sigma_2 \cdot \Sigma_3 \cdot \Sigma_4 \cdot \Sigma_5$, where $\Sigma_1, \dots, \Sigma_5$ are defined as follows.

(Σ_1) Σ_1 is the substitution that makes ground all variables in the A_i that are paired to a ground term. Z is not among these variables. Then:

1 $\text{Vardepth}(B'_i\Sigma_1) \leq \text{Vardepth}(B'_i)$ and $\text{Varnr}(B'_i\Sigma_1) \leq \text{Varnr}(B'_i)$, because Σ_1 does not affect the B'_i.

2 $\text{Vardepth}(A'_i\Sigma_1) \leq \text{Vardepth}(A'_i)$, and $\text{Varnr}(A'_i\Sigma) \leq \text{Varnr}(A'_i)$, because variables are replaced by ground terms.

(Σ_2) $\Sigma_2 = \{Z := t\}$, where t is a term of maximal Vardepth occurring in $B'_1\Sigma_1$, and Z is a $\sqsubset$-maximal variable. It must be the case that $\text{Vardepth}(t) > 0$, $\text{Vardepth}(t) = \text{Vardepth}(B'_1\Sigma_1) = \text{Vardepth}(B'_1)$, and $\text{Vardepth}(B'_1) > 0$ by assumption. Because of this t contains all variables of $B'_1 = B'_1\Sigma_1$. Σ_2 does not affect any of the $B'_i\Sigma_1$, because Z occurs only in the A'_i. We now have

1 $\text{Var}(B'_1\Sigma_1\Sigma_2) \subseteq \text{Var}(A'_i\Sigma_1\Sigma_2)$, because every $A'_i\Sigma_1\Sigma_2$ contains t.
2 $\text{Vardepth}(A'_i\Sigma_1\Sigma_2) = \text{Vardepth}(B'_1)$, because t is the only non-ground and functional term in $A'_i\Sigma_1\Sigma_2$.
3 $\text{Vardepth}(B'_i\Sigma_1\Sigma_2) \leq \text{Vardepth}(A'_i\Sigma_1\Sigma_2) = \text{Vardepth}(B'_1)$, because $\text{Vardepth}(B'_i\Sigma_1\Sigma_2) = \text{Vardepth}(B'_i)$.

(Σ_3) Σ_3 is the unifier of t with the remaining terms with which t is paired. These are the terms with which Z was paired. Since they are weakly covering, and maximal in the B'_i, we have the following:

1 $\text{Vardepth}(A'_i\Sigma_1\Sigma_2\Sigma_3) \leq \text{Vardepth}(A'_i\Sigma_1\Sigma_2)$. This follows from Theorem 2.10,
2 $\text{Vardepth}(B'_i\Sigma_1\Sigma_2\Sigma_3) = \text{Vardepth}(t\Sigma_1\Sigma_2\Sigma_3)$. This follows from Theorem 2.10, and the fact that the terms with which t is paired are the terms with maximal Vardepth.
3 $\text{Var}(B'_i\Sigma_1\Sigma_2\Sigma_3) \subseteq \text{Var}(B'_1\Sigma_1\Sigma_2\Sigma_3)$.

(Σ_4) Σ_4 is a substitution that replaces each of the remaining variables in the A'_i by one of the terms with which it is paired. We have

$$\text{Var}(A'_i\Sigma_1\Sigma_2\Sigma_3\Sigma_4) = \text{Var}(B'_i\Sigma_1\Sigma_2\Sigma_3\Sigma_4)$$

and

$$\text{Vardepth}(A'_1\Sigma_1\Sigma_2\Sigma_3\Sigma_4) \leq \text{Vardepth}(B'_1).$$

(Σ_5) Σ_5 is the remaining unification. Since Σ_5 unifies terms with the same set of variables, Σ_5 must assign either a variable, or a ground term to each variable, hence the depth cannot increase.

The result follows by collecting all the inclusions and inequalities. □

Now that we have Theorem 4.7, we can define the strategy that we described in the introduction:

DEFINITION 4.8. *The decision procedure consists of the following derivation rules:*

1 Let c be a clause. If c has a factor, then the construction of this factor is always allowed.
2 Let $c_1 = \{A_1\} \cup R_1$ and $c_2 = \{\neg A_2\} \cup R_2$ be clauses such that A_1 and A_2 are unifiable. Construction of the resolvent is allowed if for each $i = 1, 2$ one of the following holds:

(a) c_i is ground, or
(b) c_i contains non-ground functional terms, and $\text{Vardepth}(A_i)$ *is maximal in c_i.*

3 *Let c be non-ground and without functional terms. Write*

$$c = \{\neg A_1, \dots, \neg A_n\} \cup R,$$

where $\neg A_1, \dots, \neg A_n$ is a loose guard. If there are n clauses

$$c_1 = \{B_1\} \cup R_1, \dots, c_n = \{B_n\} \cup R_n,$$

such that either

(a) for each i, either c_i is ground or
(b) c_i contains non-ground functional terms, and $\mathrm{Vardepth}(B_i)$ *is maximal in c_i,*

and a hyperresolvent is possible, then construct a permutation $(\pi_1, \dots, \pi_n)$, and an m as in Theorem 4.7. Write

$$(A'_1, \dots, A'_n) = (A_{\pi_1}, \dots, A_{\pi_n}),$$

$$(B'_1, \dots, B'_n) = (B_{\pi_1}, \dots, B_{\pi_n}),$$

$$(R'_1, \dots, R'_n) = (R_{\pi_1}, \dots, R_{\pi_n}),$$

and construct a partial hyperresolvent as follows: From

$$\{\neg A'_1, \dots, \neg A'_m, \neg A'_{m+1}, \dots, \neg A'_n\} \cup R$$

and

$$\{B'_1\} \cup R'_1, \dots, \{B'_m\} \cup R'_m$$

construct

$$\{\neg A'_{m+1}\Theta', \dots, \neg A'_n\Theta'\} \cup R\Theta' \cup R'_1\Theta' \cup \dots \cup R'_m\Theta'.$$

Making use of Theorem 4.7, the termination proof is analogous to the termination proof for the guarded fragment.

LEMMA 4.9. *Let c be a loosely guarded clause. Let Θ be a substitution that does not assign a non-ground functional term to any variable. Then $c\Theta$ is loosely guarded. Moreover, for every set of literals $G \subseteq c$ that form a loose guard of c, the instantiation $G\Theta$ is a loose guard of $c\Theta$.*

THEOREM 4.10. *Let C be a loosely guarded clause set, let $v =$* $\mathrm{Vardepth}(C)$. *Every clause that is derivable by the refinement of Definition 4.8 is loosely guarded, does not have a* Vardepth *greater than v, and has a loose guard, that is an instance of a loose guard in a clause of C.*

PROOF. 1 Suppose that c has been obtained by factoring from a parent clause c_1. It follows in the same way as in the proof of Theorem 3.8, that the substitution Θ does not assign a non-ground, functional term to a variable in $c_1\Theta$. Then Lemma 4.9 can be applied, to obtain that c is loosely guarded and has a loose guard that is an instance of a loose guard in c_1. It follows immediately from the fact that Θ does not assign non-ground functional terms that $\mathrm{Vardepth}(c_1\Theta) \leq \mathrm{Vardepth}(c)$.

2 Let c be obtained from c_1 and c_2 by binary resolution, using an mgu Θ. One can show in essentially the same way as in the proof of Theorem 3.8 that each non-ground, functional term in c contains all variables of c, and that $\mathrm{Vardepth}(c) \leq$

Vardepth(c_1) or Vardepth(c) $\leq$ Vardepth(c_2). One also obtains that for one of c_1, c_2 the following holds: The substitution Θ does not assign a non-ground, functional term to any of the variables in c_i. This ensures that c has a loose guard that is an instance of a loose guard of c_i.

3 Let

$$h = \neg G_1\Theta \cup \cdots \cup \neg G_m\Theta \cup \{\neg A_{m+1}\Theta, \ldots, \neg A_n\Theta\} \cup R\Theta \cup R_1\Theta \cup \cdots \cup R_m\Theta$$

be obtained by partial hyperresolution from the following loosely guarded clauses:

$$c = \{\neg A_1, \ldots, \neg A_m\} \cup \{\neg A_{m+1}, \ldots, \neg A_n\} \cup R,$$

$$c_1 = \neg G_1 \cup R_1 \cup \{B_1\},$$

$$\ldots$$

$$c_m = \neg G_m \cup R_m \cup \{B_m\}.$$

with substitution Θ. The $\neg G_i$ are the loose guards of clauses c_i. We will show that $\neg G_1\Theta$ is a loose guard of h. From Theorem 4.7, Part 1, we know that Θ does not assign a non-ground functional term to a variable in c_1. Therefore we can apply Lemma 4.9 and we know that $\neg G_1\Theta \cup R_1\Theta \cup \{B_1\Theta\}$ is a loosely guarded clause, with loose guard $\neg G_1\Theta$. Now all the B_i contain all variables of their clauses c_i. From Theorem 4.7, Part 2, it follows that $\mathrm{Var}(B_i\Theta) \subseteq \mathrm{Var}(B_1\Theta)$. This makes sure that $\neg G_1\Theta$ is a loose guard of h.

Next we must show that every non-ground functional term in h contains all variables of h. Let t be a non-ground functional term in h. First consider the case where t originates from one of the parents c_i. If there is a variable V in c_i, such that $V\Theta = t$, then this variable occurs in B_i. Since $B_i\Theta$ is weakly covering (by Theorem 4.7, Part 4), the result $V\Theta = t$ contains all variables of h. If there is a term u in c_i, such that $u\Theta = t$, then this term contains all variables of c_i. Hence $u\Theta = t$ contains all variables of $c_i\Theta$. The case where t originates from c is completely analogous.

Finally we show that $\mathrm{Var}(h) \subseteq \mathrm{Var}(c_1)$ and $\mathrm{Vardepth}(h) \subseteq \mathrm{Vardepth}(c_1)$. We originally have

$$\mathrm{Var}(c_i) \subseteq \mathrm{Var}(B_i),\ \mathrm{Vardepth}(c_i) \leq \mathrm{Vardepth}(B_i).$$

This implies that

$$\mathrm{Var}(c_i\Theta) \subseteq \mathrm{Var}(B_i\Theta),\ \mathrm{Vardepth}(c_i\Theta) \leq \mathrm{Vardepth}(B_i\Theta).$$

From Theorem 4.7, Part 2, we have

$$\mathrm{Var}(B_i\Theta) \subseteq \mathrm{Var}(B_1\Theta),\ \mathrm{Vardepth}(B_i\Theta) \leq \mathrm{Vardepth}(B_1\Theta).$$

Combining this and applying Part 1 of Theorem 4.7 completes the proof.

□

It remains to show that the set of derivable clauses is finite and to obtain a complexity bound. One can prove the analog of Lemma 3.10 in essentially the same way. This makes it possible to apply Theorem 3.22 with the following modification: In point **(1)**, one has to replace 'the maximal arity of a guard', by 'the maximal number of variables in a loose guard'.

4.3. Completeness

The strategy for the loosely guarded fragment is more complex than the strategy for the guarded fragment. The strategy is also non-liftable, but moreover, it does not have a natural definition that uses orders. In order to prove its completeness we need to modify the resolution game, such that it can handle the partial hyperresolution rule.
The closest existing approximation of what we need is *A-ordered resolution with selection*, that occurs in (Bachmair and Ganzinger, 1994). We repeat the definition here.

Definition 4.11. *Let c be a set of propositional clauses. Let $\sqsubset$ be an order on atoms. Extend $\sqsubset$ to literals as follows:*

$$A \sqsubset B \textit{ implies } \neg A, A \sqsubset \neg B, B.$$

Let σ be a function from sets of literals to sets of literals satisfying:

1 $\sigma(c) \subseteq c$, for each clause c.
2 For each clause c, either $\sigma(c)$ contains all $\sqsubset$-maximal literals, or $\sigma(c)$ contains at least one negative literal.

Having the selection function, when we construct the resolvent

$$\{\neg A\} \cup R_1, \{A\} \cup R_2 \Rightarrow R_1 \cup R_2,$$

we impose that condition that

$$\neg A \in \sigma(\{\neg A\} \cup R_1), \quad A \in \sigma(\{A\} \cup R_2).$$

Example 4.12. Assume that $a \sqsubset b$. Look at the clause $c = \{a, b, \neg a, \neg b\}$. It is allowed to have $\sigma(c) = \{b\}$. It is not allowed to have $\sigma(c) = \{a\}$. It is allowed to have $\sigma(c) = \{\neg a\}$, or $\sigma(c) = \{\neg b\}$.

It is not required to select a single literal, so it is allowed to have $\sigma(c) = \{a, b\}, \sigma(c) = \{\neg a, b\}$. In the propositional case, that we have defined here, it is always possible to make $\sigma(c)$ a singleton. Hyperresolution can be seen as a special form of resolution with selection, by always selecting exactly one negative literal, if there is one. Standard A-ordered resolution can be obtained by always selecting consistent with $\sqsubset$.
It is shown in (Bachmair and Ganzinger, 1994) that this restriction of resolution is complete, and that it can be combined with certain restrictions of paramodulation. The relation to our strategy can best be explained by using Example 4.5. We would like to use selection on clause c_0 to select the literals $\neg a_1(X, Y), \neg a_2(Y, Z), \neg a_3(Z, X)$, but this is not possible, because it depends on the clauses c_1, c_2, c_3, which literals of the loosely guard should be resolved away. There might be different clauses c_1', c_2', c_3', for which other literals should be selected. However in the completeness proof of resolution with selection functions, the fact that the selection is made in advance, is not used. All that is used there is that, if there is a clause $\{\neg a_1, \ldots, \neg a_p\} \cup R$ with one of the literals $\neg a_1, \ldots, \neg a_p$ selected, and for each i there is a clause of the form $\{a_i\} \cup R_i$, with a_i selected, then there is at least one clause of the form $\{\neg a_1, \ldots, \neg a_{i-1}, \neg a_{i+1}, \ldots, \neg a_p\} \cup R \cup R_i$, for some i. This can be ensured by selecting a fixed literal from the $\neg a_1, \ldots, \neg a_p$ in advance, but it is not necessary. So we need a generalization of the results in (Bachmair and

Ganzinger, 1994), with a non-liftable order, and without having to make the selection in advance. For this we need to adapt the resolution game.

DEFINITION 4.13. *We define the new resolution game as an ordered quadruple* $\mathcal{G} = (P, \mathcal{PA}, \prec, \sigma)$. *Here* P *is a set of propositional atoms, as before.* $\mathcal{PA}$ *is a set of indexed atoms. It is not required that all pairs of a propositional symbol and an attribute do occur in* $\mathcal{PA}$. Literals *and* indexed literals *are as before. The order* $\prec$ *is well-founded as before, but it is defined on* $\mathcal{PA}$ *instead of* $(P \cup \neg P) \times \mathcal{A}$. *It is extended to indexed literals by*

$$a{:}A \prec b{:}B \Rightarrow \pm{:}aA \prec b{:}\pm B.$$

A clause *is a structure of the form* $c_g \vdash c_r$. *Here* c_g *is a finite multiset of atoms, and* c_r *is a finite multiset of indexed literals.*
For a clause $c_g \vdash c_r$, *the* selection function *equals either* c_g *or* c_r. *If* $\sigma(c_g \vdash c_r) = c_g$, *we say that* c_g *is selected. In the other case we say that* c_r *is selected. If* c_r *is selected, the clause* $c_g \vdash c_r$ *can be used for binary resolution and factoring. If* c_g *is selected, the clause* $c_g \vdash c_r$ *can be used for partial hyperresolution and factoring.*
If c_r *is selected, then it must be the case that for every atom* a *in* c_g, *and for all indexed literals* $a{:}A$ *that can be built using* a, *there is an indexed literal* $b{:}B$ *in* c_r, *such that* $a{:}A \prec b{:}B$.
We have the following condition on atoms that occur in the left hand side: If an atom a *occurs in the left hand side of a clause* $c_g \vdash c_r$, *then there exists an* $a{:}A \in \mathcal{PA}$, *such that for all other* $a{:}A'$, *based on* a, *it is the case that* $a{:}A' \prec a{:}A$.
Reductions *are obtained by finitely often making the following replacements.*

1 *Replacing* $c_g \cup [a] \vdash c_r$ *by* $c_g \vdash c_r \cup [\neg a{:}A]$.
2 *Replacing* $c_g \vdash c_r \cup [a{:}A]$ *by some* $c_g \vdash c_r \cup [a{:}A']$ *with* $a{:}A' \prec a{:}A$.

The modified resolution game has the following derivation rules:

FACTOR 1 *If a clause* c_1 *has form* $c_g \vdash [b{:}B_1, b{:}B_2] \cup R$, *and the right hand side is selected, and* $b{:}B_1$ *is maximal, then* $c_g \vdash [b{:}B_1] \cup R$ *is a factor of* c_1.
2 *If a clause* c_1 *has form* $[a] \cup c_g \vdash [\neg a{:}A] \cup R$, *the right hand side is selected, and* $\neg a{:}A$ *is maximal, then* $[a] \cup c_g \vdash R$ *is a factor of* c_1.

RES *If* $c_1 \vdash R_1 \cup [b{:}B_1]$, *and* $c_2 \vdash R_2 \cup [\neg b{:}B_2]$ *are clauses with their right hand sides selected, and* $b{:}B_1$ *and* $\neg b{:}B_2$ *are maximal in their clauses, then the following clause is a resolvent:*

$$c_1 \cup c_2 \vdash R_1 \cup R_2.$$

PARTIAL *Let*

$$r = [a_1, \ldots, a_p] \vdash R$$

be a clause, such that the left hand side $[a_1{:}A_1, \ldots a_p{:}A_p]$ *is selected. Let*

$$g_1 \vdash [a_1{:}A'_1] \cup R_1, \quad \ldots, \quad g_p \vdash [a_p{:}A'_p] \cup R_p$$

be clauses, such that all $a_i{:}A'_i$ *are maximal in their clauses, and all* $[a_i{:}A'_i] \cup R_i$ *are selected. Let* $m \leq p$. *Then clauses of the following form are* partial hyperresolvents*:*

$$g_1 \cup \cdots \cup g_m \cup [a_{m+1}, \ldots, a_p] \vdash R \cup R_1 \cup \cdots \cup R_m.$$

(We have omitted the permutation for notational reasons)

DEFINITION 4.14. *Let C be a set of clauses. A* saturation *$\overline{C}$ of C is a set of clauses satisfying the following:*

1 *$C \subseteq \overline{C}$.*
2 *For every clause $c_g \vdash c_r$ that can be obtained from clauses in $\overline{C}$, either by RES, or by FACTOR, there is a reduction $d_g \vdash d_r$ of $c_g \vdash c_r$ in $\overline{C}$.*
3 *For every group of clauses $r; c_1, \ldots, c_n$, such that it is possible to form partial hyperresolvents, there is at least one reduction $d_g \vdash d_r$ of one of the partial hyperresolvents in $\overline{C}$.*

We have the following completeness theorem:

THEOREM 4.15. *Let $\overline{C}$ be a saturation of a clause set C. If $\overline{C}$ does not contain the empty clause, then C has a model.*

PROOF. Assume that a saturated clause set $\overline{C}$ does not contain the empty clause. We show that $\overline{C}$ has a model. The order $\prec$ of the resolution game is well-founded on $\mathcal{PA}$. Without loss of generality we can assume that $\prec$ is total. Let k be the ordinal of the length of $\prec$. We inductively construct sets $I_0, I_1, \ldots, I_\omega, \ldots$ up to I_k as follows:

1 $I_0 = \{\ \}$.
2 For a successor ordinal $\lambda + 1$, let $b{:}B$ be the indexed literal on position λ.
   (a) Put $I_{\lambda+1} = I_\lambda \cup \{b{:}B\}$ if either there is a reduction $b{:}B'$ of $b{:}B$ in I_λ, or there is a clause c in $\overline{C}$ which has form
   $$c = [a_1, \ldots, a_p] \vdash r{:}R \cup [b{:}B],$$
   such that
      i the right hand side of c is selected,
      ii c cannot be factored,
      iii $b{:}B$ is the maximal indexed literal in c,
      iv for each literal a_i of the left hand side of c_g, there is an indexed literal $a{:}A \in I_\lambda$,
      v there is no literal in $r{:}R$, that occurs in I_λ.
   (b) Put $I_{\lambda+1} = I_\lambda \cup \{\neg b{:}B\}$ on the same conditions as for $b{:}B$, but with $b{:}B$ replaced by $\neg b{:}B$.
   (c) Otherwise put $I_{\lambda+1} = I_\lambda$.

   Observe that Cases 1 and 2 may overlap. When that happens, we assume that Case 1 is checked before Case 2. Because of this, $b{:}B$ is added, and $\neg b{:}B$ is not added.
3 For a limit ordinal λ, put $I_\lambda = \bigcup_{\mu<\lambda} I_\mu$.

We first establish the following property:

JUST For each indexed literal $\pm b{:}B$ in I_k, there is a clause of the form $c = [a_1, \ldots, a_p] \vdash r{:}R \cup [\pm b{:}B]$ in $\overline{C}$, such that

   1 The right hand side of c is selected,

2 c cannot be factored,
3 $\pm b\colon B$ is the maximal indexed literal of c,
4 For each a_i, there is an indexed literal of the form $a_i\colon A_i \in I_k$,
5 No literal of $r\colon R$ is in I_k.

The problem is to establish (5). It is clearly the case that no literal of $r\colon R$ occurs in I_λ, because of Condition v of the construction. The indexed literals $\pm a\colon A$, that are added later, all have $\pm b\colon B \prec \pm a\colon A$. Since $\pm b\colon B$ is the maximal literal of c, they cannot be in c. Next we will show the following two facts by induction:

A for indexed atoms $a\colon A$, it is not the case that both $a\colon A$ and $\neg a\colon A$ are in I_k,

and for each clause $c_g \vdash c_r$ in $\overline{C}$, at least one of the following is true:

C1 For an a in c_g, there is no A, such that $a\colon A \in I_k$.
C2 There is an $a\colon A$ in c_r, such that $a\colon A$ in I_k.
C3 There is a $\neg a\colon A$ in c_r, such that $\neg a\colon A$ in I_k.

We write C for the disjunction C1 $\vee$ C2 $\vee$ C3.
We will establish A and C by induction on the multiset extension $\prec\!\prec$ of $\prec$. In order to do this we associate a finite multiset of indexed atoms to each instance of A and C as follows:

1 To A, applied to an indexed atom $a\colon A$, we associate the multiset $[a\colon A]$.
2 To C, applied to a clause $[a_1, \ldots, a_p] \vdash c_g$ we associate the multiset $[a_1\colon A_1, \ldots, a_p\colon A_p] \cup c_g$. Here each $a_i\colon A_i$ is the maximal indexed atom that can be constructed from a_i.

In the induction proof we need the following property:

REDUCTION Let S be a finite multiset of indexed literals. Suppose that we have already established the induction hypotheses to all finite multisets below S. Let $c_g \vdash c_r$ be some clause, not necessarily in $\overline{C}$, with associated multiset below S. Let $d_g \vdash d_r$ be a reduction of $c_g \vdash c_r$ that occurs in $\overline{C}$. Then $c_g \vdash c_r$ also satisfies C.

First observe that $d_g \vdash d_r$ also has the associated multiset below S. It is sufficient to show that REDUCTION is preserved by reductions that consist of one step.

1 Consider the case where $c_g \vdash c_r \cup [\neg a\colon A]$ is a reduction of $c_g \cup [a] \vdash c_r$. Assume that $c_g \vdash [a] \cup [\neg a\colon A]$ satisfies one of C1, C2, C3. If $c_g \vdash c_r \cup [\neg a\colon A]$ satisfies C1, then $c_g \cup [a] \vdash c_r$ also satisfies C1. If $c_g \vdash c_r \cup [\neg a\colon A]$ satisfies one of C1, C2, then one of the literals in $c_r \cup [\neg a\colon A]$ occurs in I_k. If this literal is in c_r, then $c_g \cup [a] \vdash c_r$ clearly satisfies one of C1, C2. If it is $\neg a\colon A$, then let $\neg a\colon A'$ be the maximal indexed literal based on a. By the construction of I_k, it must be the case that $\neg a\colon A' \in I_k$. The associated multiset $[a\colon A'] \prec\!\prec$ the associated multiset of $c_g \cup [a] \vdash c_r$. Hence we can apply A to obtain that $a\colon A'$ is not in I_k. This makes that $c_g \cup [a] \vdash c_r$ satisfies C1.
2 Consider the case where $c_g \vdash c_r \cup [\pm a\colon A']$ is a reduction of $c_g \vdash c_r \cup [\pm a\colon A]$. If $c_g \vdash c_r \cup [\pm a\colon A']$ satisfies C1, then $c_g \vdash c_r \cup [\pm a\colon A]$ also satisfies C1. If $c_g \vdash c_r \cup [\pm a\colon A']$

satisfies one of C2, C3, and a literal of c_r is in I_k, then clearly $c_r \vdash c_r \cup [\pm a{:}\,A]$ satisfies one of C2, C3. If $c_g \vdash c_r \cup [\pm a{:}\,A']$ satisfies one of C2, C3, and $\pm a{:}\,A'$ is in I_k, then by the construction of I_k, also $\pm a{:}\,A \in I_k$. Hence $c_g \vdash c_r \cup [a{:}\,A]$ satisfies one of C2, C3.

Let S be a finite multiset of indexed atoms. Assume that A and C are true for all instances with associated multiset below S. We prove that instances of A and C with associated multiset equal to S are also true. We do this by analyzing the possible instances that have an associated multiset S. More than one case can be applicable, and it is possible that no case applies.

1 If S has the form $[a{:}\,A]$, then we have to establish the fact that not both $a{:}\,A$ and $\neg a{:}\,A$ are in $\overline{C}$. Suppose that they were both in $\overline{C}$. Then there are clauses

$$c_1 = c_g^1 \vdash c_r^1 \cup [a{:}\,A], \text{ and } c_2 = c_g^2 \vdash c_r^2 \cup [\neg a{:}\,A]$$

in $\overline{C}$ satisfying JUST. The resolvent $c_g^1 \cup c_g^2 \vdash c_r^1 \cup c_r^2$ is allowed, and therefore a reduction $d_g \vdash d_r$ of it is in $\overline{C}$. Now the resolvent has an associated multiset smaller than S, because it consists of indexed literals strictly below $a{:}\,A$. We can apply REDUCTION, and we obtain the fact that the resolvent $c_g^1 \cup c_g^2 \vdash c_r^1 \cup c_r^2$ satisfies C. We show that this leads to a contraction. If the resolvent satisfies C1, this means that for one of the atoms a in $c_g^1 \cup c_g^2$, there is no indexed atom $a{:}\,A \in I_k$ This makes that one of the clauses c_1, c_2 violates Condition 4 of JUST. If the resolvent satisfies C2 or C3 this leads to a violation of Condition 5 of JUST in the same way.

2 If there is a clause of the form $c = [a_1, \ldots, a_p] \vdash R$ in $\overline{C}$, with the left hand side selected, and with associated multiset S, then assume that c does not satisfy C1. We will show that c satisfies either C2 or C3. There must exist clauses

$$g_1 \vdash [a_1{:}\,A_1] \cup R_1, \ldots, g_p \vdash [a_p{:}\,A_p] \cup R_p$$

in $\overline{C}$, that satisfy JUST. Because of this a partial hyperresolvent is possible. Assume that there is a reduction of the partial hyperresolvent

$$h = g_1 \cup \cdots \cup g_m \cup [a_{m+1}, \ldots, a_p] \vdash R \cup R_1 \cup \cdots \cup R_m.$$

The associated multiset of h is smaller than S. This is because in the clauses $g_i \vdash [a_i{:}\,A_i] \cup R_i$, all indexed literals in R_i are strictly smaller than $a_i{:}\,A_i$. By the conditions on selection of the right hand side, the maximal indexed atoms that can be built from g_i are strictly smaller than $a_i{:}\,A_i$. Each indexed atom $a_i{:}\,A_i$ is less than, or equal to the maximal indexed atom that can be built from a_i. This implies that the associated multiset of h can be obtained from the associated multiset of c, by replacing some indexed literals by a finite set of strictly smaller indexed literals. Because of this we can apply REDUCTION, and we obtain the fact that h satisfies C. We can proceed in essentially the same way as in the previous case. We first show that h must satisfy C2 or C3, because C1 results in a contradiction. Suppose that h satisfies C1. If for one of the $a_2, \ldots, a_p$, there is no A_i, such that $a_i{:}\,A_i \in I_k$, this contradicts the initial assumption. If for an atom a in one of the g_i, there is no indexed atom $a{:}\,A \in I_k$, this constradicts Condition 4 of JUST. Now the fact that h satisfies C2 or C3 means that there is an indexed literal $\pm a{:}\,A$ that occurs in both $R \cup R_1 \cup \cdots \cup R_m$ and I_k. Because each $g_i \vdash [a_i{:}\,A_i] \cup R_i$ satisfies Condition 5 of

JUST, the only possibility is that the indexed literal $\pm a\colon A$ occurs in R. This makes that c satisfies C2 or C3.

3 If there is a clause of the form $c_g \vdash c_r$ in $\overline{C}$, with the right hand side selected, which can be factored and with associated multiset S, then we write $c'_g \vdash c'_r$ for one of its factors, and let $d_g \vdash c_r$ be a reduction that is in $\overline{C}$. It is easily checked that both have an associated multiset strictly smaller than S, and because of this we can apply REDUCTION and obtain that $c'_g \vdash c'_r$ satisfies C. Then it is easily checked that $c_g \vdash c_r$ satisfies C.

4 If there is a clause of the form $c_g \vdash c_r$, with the right hand side selected, which cannot be factored and with associated multiset S, then proceed as follows: Suppose that c_g does not satisfy C1. Let $\pm a\colon A$ be the (unique) maximal literal in $c_g \vdash c_r$. Let λ be its position in the ordering. Then at the moment that $I_{\lambda+1}$ was constructed there already was an indexed literal $c\colon C \in I_\lambda$, for each $c \in c_g$. (Because the right hand side of $c_g \vdash c_r$ was selected, there do not exist indexed literals $c\colon C$ with $c \in c_g$ greater than $\pm a\colon A$.) If at the moment that $I_{\lambda+1}$ was constructed, $c_g \vdash c_r$ did not satisfy C2 or C3, then $\pm a\colon A$ is added to $I_{\lambda+1}$. For this reason $c_g \vdash c_r$ necessarily satisfies C2 or C3.

Finally, a model of $\overline{C}$ can be extracted from I_k by putting the atoms a, for which there is an indexed atom $a\colon A$ in I_k, true. The other atoms are put false. It follows from A, C1, C2, C3, that this makes every clause in $\overline{C}$ true. □

DEFINITION 4.16. *Let A be literal. The* normalization *of A is defined as in Definition 3.14, but if A is negative, the negation sign is removed in the process.*
Let $c = \{\neg a_1\colon A_1, \ldots, \neg a_p\colon A_p, b_1\colon B_1, \ldots, b_q\colon B_q\}$ be a representation-indexed, loosely guarded clause with loose guard $\{\neg a_1\colon A_1, \ldots, \neg a_p\colon A_p\}$. Let Θ be its substitution. Let $k = \#\Theta$. Then $[c]$ is defined as

$$[a_1, \ldots, a_p] \vdash [b_1\colon (k, \overline{B_1}), \ldots, b_q\colon (k, \overline{B_q})].$$

Here the $\overline{A_i}, \overline{B_i}$ are the normalizations of the A_i, B_i.

THEOREM 4.17. *The strategy of Definition 4.8 is complete for clause sets C in the loosely guarded fragment.*

PROOF. Once we have the resolution game of Definition 4.13, the proof is analogous to the proof of Theorem 3.19. Let C be an unsatisfiable, loosely guarded clause set. Let $\overline{C}$ be its closure under resolution and factoring, using the rules of Definition 4.8. We need to show that $\overline{C}$ contains the empty clause. Let C_{hb} and $\overline{C}_{hb}$ be obtained as in Theorem 3.19. The set of propositional symbols P is defined as the set of propositional atoms in C_{hb}. The set $[\overline{C}_{hb}]$ is defined as before, but using the new definition of [ ], given in Definition 4.16. The set $\mathcal{PA}$ is defined as the set of objects $a\colon (k, \overline{A})$ for which either $a\colon (k, \overline{A})$ or $\neg a\colon (k, \overline{A})$ occurs in $[\overline{C}_{hb}]$.
The selection function σ is defined as follows: Let

$$c = [a_1, \ldots, a_p] \vdash [b_1\colon (k, \overline{B_1}), \ldots, b_q\colon (k, \overline{B_q})]$$

be a clause in $[\overline{C}_{hb}]$. If there is an indexed literal $b_j\colon (k, \overline{B_j})$ containing non-ground, functional terms, then select the right hand side of c. Otherwise select the left hand side. We must show that when the right hand side is selected, the clause satisfies the condition in Definition 4.13. Because the $a_i\colon(k, A_i)$ are part of the loose guard, they do not contain non-ground, functional terms. Let Σ be the substitution such that $a_i = A_i\Sigma$. Because A_i does not contain non-ground, functional terms, there exist no term A' and Σ', such that $a_i = A'\Sigma'$, and $\#\Sigma < \#\Sigma'$, so we know that $\#\Sigma \geq \#\Sigma'$. We also have $\#\Sigma \leq k$. (They are not necessarily equal because A_i need not contain all variables in the clause) From this it follows that there are no indexed atoms $a_i\colon(l, A_i') \in [\overline{C}_{hb}]$ with $k < l$, or $k = l$, and A_i' contains non-ground, functional terms.

We also need to show that for every atom occurring in a guard, there is a maximal indexed atom, based on a in $\mathcal{PA}$. This is the case because $\mathcal{PA}$ is finite.

It remains to show that $[\overline{C}_{hb}]$ is a saturation of the resolution game. This is essentially analogous to the proof of Theorem 3.19. The differences are the following:

When, due to substitution, a literal moves from the loose guard to the body of a clause, this is modelled by the first type of reduction, in Definition 4.13.

When a partial hyperresolvent is formed, assume that $[a_1, \ldots, a_p] \vdash r\colon(k, R)$ and

$$g_1 \vdash [a_1\colon(k_1, A_1)] \cup r_1\colon(k_1, R_1),$$

$$\ldots$$

$$g_p \vdash [a_p\colon(k_p, A_p)] \cup r_p\colon(k_p, R_p)$$

have a partial hyperresolvent. There must exist clauses of the following form in $\overline{C}_{hb}$,

$$c = \{\neg a_1\colon A_1, \ldots, \neg a_p\colon A_p\} \cup r\colon R,$$

$$c_1 = \{\neg g_1\colon G_1\} \cup r_1\colon R_1 \cup \{a_1\colon A_1\},$$

$$\ldots$$

$$c_m = \{\neg g_m\colon G_m\} \cup r_m\colon R_m \cup \{a_m\colon A_m\},$$

$$\ldots$$

$$c_p = \{\neg g_p\colon G_p\} \cup r_p\colon R_p \cup \{a_p\colon A_p\}.$$

with partial hyperresolvent $h =$

$$\neg g_1\colon G_1\Theta \cup \cdots \cup \neg g_m\colon G_m\Theta \cup \{\neg a_{m+1}\colon A_{m+1}\Theta, \ldots, \neg a_p\colon A_p\Theta\}$$

$$\cup r\colon R\Theta \cup r_1\colon R_1\Theta \cup \cdots \cup r_m\colon R_m\Theta.$$

Write $[h] =$

$$g_1 \cup \cdots \cup g_m \cup [a_{m+1}, \ldots, a_p] \vdash r\colon(l, R\Theta) \cup r_1\colon(l, R_1\Theta) \cup \cdots \cup r_p\colon(l, R_p\Theta).$$

It is sufficient to show that $[h]$ is a reduction of the following partial hyperresolvent

$$g_1 \cup \cdots \cup g_m \cup [a_{m+1}, \ldots, a_p] \vdash r\colon(k, R) \cup r_1\colon(k_1, R_1) \cup \cdots \cup r_p\colon(k_p, R_p).$$

This is essentially analogous to the proof of Theorem 3.19. It is sufficient to prove that $l \leq k_i$, and $l \leq k$. This follows from the fact that for each $i, 1 \leq i \leq m$,

$$\mathrm{Var}(c_i\Theta) \subseteq \mathrm{Var}(c\Theta).$$

□

THEOREM 4.18. *Resolution with factoring, as defined in Definition 4.8, together with the modified normal form transformation, is a decision procedure for the loosely guarded fragment.*

5. Conclusions and Further Work

We have shown that it is possible to effectively decide the guarded fragment and the loosely guarded fragments by resolution. The proofs that the resolution refinements are complete and terminating can be used as proofs for the decidability of these fragments, but they offer more than that. They also define practical decision procedures, using techniques that are standard to the theorem proving community. This has made implementation relatively easy. Since the procedures could be built on top of an existing resolution prover, they could easily be combined with an efficient, full first order theorem prover (de Nivelle, 1999a)

Our decision procedure has interest in itself, but it can also be applied to modal logics, using the relational translation. From the space point of view, translation into the guarded fragment is not the optimal way for deciding simple modal logics like K and T, since these logics are in PSPACE (Ladner, 1977), while the complexity of the guarded fragment with fixed arity is single exponential. However it is not likely that a resolution decision procedure will ever decide modal logics in PSPACE, since resolution cannot even solve propositional logic in PSPACE.

We expect that our methods has advantages over the direct approaches of resolution in modal logic (Enjalbert and Fariñas del Cerro, 1989; de Nivelle, 1993), because our method provides a decision procedure, and because it can exploit existing implementations.

We do not expect to be able to improve the functional translation methods (Schmidt, 1997), at least not with our present translation.

A natural question is, whether or not the results in (Grädel and Walukiewicz, 1999) can be obtained by resolution. We are pessimistic but we will investigate the question.

Acknowledgments.

Maarten de Rijke and Hans de Nivelle were both supported by the Spinoza Project 'Logic in Action' at ILLC, the University of Amsterdam. Hans de Nivelle was also supported by NWO.

References

Andréka, H., van Benthem, J., Németi, I. (1998). Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, **27**:217–274.

Baaz, M., Fermüller, C., Leitsch, A. (1994). A non-elementary speed up in proof length by structural clause form transformation. In *Proceedings LICS'94*, pages 213–219.

Bachmair, L., Ganzinger, H. (1994). Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, **4**:217–247.

Boyer, R. (1971). *Locking: A Restriction of Resolution*. PhD thesis, University of Texas at Austin.

Chang, C.-L., Lee, R.-T. (1973). *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.

de Nivelle, H. (1993). Generic resolution in propositional modal systems. In *LPAR'93*, volume 698 of *LNAI*, pages 241–252.

de Nivelle, H. (1994). Resolution games and non-liftable resolution orderings. In *CSL'94*, pages 279–293.

de Nivelle, H. (1998). A resolution decision procedure for the guarded fragment. In *CADE'98*, pages 191–204.

de Nivelle, H. (1999a). *The Bliksem theorem prover*. Can be obtained from `http://www.mpi-sb.mpg.de/~bliksem`.

de Nivelle, H. (1999b). Translation of S4 and K4 into the guarded fragment and the 2-variable fragment. manuscript.

de Rijke, M. (1999). Restricted description languages. Manuscript, ILLC, University of Amsterdam.

Enjalbert, P., Fariñas del Cerro, L. (1989). Modal resolution in clausal form. *Theoretical Computer Science*, **65**(1):1–33.

Fermüller, C., Leitsch, A., Tammet, T., Zamov, N. (1993). *Resolution Methods for the Decision Problem.* Number 679 in LNAI. Springer Verlag.

Gabbay, D. (1981). Expressive functional completeness in tense logic. In Mönnich, U., editor, *Aspects of Philosophical Logic*, pages 91–117. Reidel.

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

Ganzinger, H., Meyer, C., Veanes, M. (1999). The two-variable guarded fragment with transitive relations. In *LICS'99*, pages 24–34.

Grädel, E. (1997). On the restraining power of guards. To appear in the Journal of Symbolic Logic.

Grädel, E., Kolaitis, P., Vardi, M. (1997). On the decision problem for two-variable first-order logic. *The Bulletin of Symbolic Logic*, **3**(1):53–69.

Grädel, E., Walukiewicz, I. (1999). Guarded Fixed Point Logic. In *LICS'99, Trento*, pages 45–54.

Immerman, N., Kozen, D. (1989). Definability with a bounded number of bound variables. *Information and Computation*, **83**:121–139.

Ladner, R. (1977). The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, **6**:467–480.

Leitsch, A. (1997). *The Resolution Calculus.* Texts in Theoretical Computer Science. Springer Verlag.

McCune, W. (1995). *Otter 3.0 Reference Manual and Guide.* Argonne National Laboratory, Mathematics and Computer Science Division. Available from `ftp.mcs.anl.gov`, directory `pub/Otter`.

Mortimer, M. (1975). On languages with two variables. *Zeitschrift für math. Logik und Grundlagen der. Math.*, **21**:135–140.

Ohlbach, H.-J., Weidenbach, C. (1995). A note on assumptions about skolem functions. *Journal of Automated Reasoning*, **15**:267–275.

Schmidt, R. (1997). *Optimized Modal Translation and Resolution.* PhD thesis, Max Planck Institut für Informatik, Saarbrücken.

Tammet, T. (1990). The resolution program, able to decide some solvable classes. In *Proceedings COLOG-88*, pages 300–312.

van Benthem, J. (1997). Dynamic bits and pieces. Technical Report LP-97-01, ILLC, University of Amsterdam.

Vardi, M. (1997). Why is modal logic so robustly decidable? In Immerman, N., Kolaitis, P., editors, *Descriptive Complexity and Finite Models*, pages 149–183.

Weidenbach, C. (1997). *The Spass & Flotter Users Guide, Version 0.55.* Available from `ftp.mpi-sb.mpg.de`, directory `pub/SPASS`.