A Superposition Decision Procedure for the Guarded Fragment with Equality

Harald Ganzinger[*]
Hans de Nivelle[†]
Max-Planck-Institut für Informatik, D-66123 Saarbrücken
{hg|nivelle}@mpi-sb.mpg.de

Abstract

We give a new decision procedure for the guarded fragment with equality. The procedure is based on resolution with superposition. We argue that this method will be more useful in practice than methods based on the enumeration of certain finite structures. It is surprising to see that one does not need any sophisticated simplification and redundancy elimination method to make superposition terminate on the class of clauses that is obtained from the clausification of guarded formulas. Yet the decision procedure obtained is optimal with regard to time complexity. We also show that the method can be extended to the loosely guarded fragment with equality.

Keywords: automated deduction, decidable fragments, modal and temporal logics

1 Introduction

The loosely guarded fragment was introduced in (Andréka, van Benthem & Németi 1996) as 'the modal fragment of classical logic'. It is obtained essentially by restricting quantification to the following forms:

$$\forall y[R(x,y) \rightarrow A(x,y)], \text{ and } \exists y[R(x,y) \wedge A(x,y)].$$

These forms naturally arise when modal formulae are translated into classical logic using the standard translation based on the Kripke frames. The authors showed there that the guarded fragment has many of the nice properties of modal logics. In particular it is decidable. Any decision procedure for this fragment, hence, is a decision procedure for those modal logics that can be embedded into it, for example $K, D, S3$, and B. Grädel (1997) has shown that equality can be admitted in the guarded fragment without affecting decidability. In the fragment with equality additional logics such as difference logic can be expressed (where $\Diamond A$ means A holds in a world different from the present).

de Nivelle (1998) has given a resolution decision procedure for the guarded fragment without equality. In his procedure, a non-liftable ordering is employed, and, hence, some additional and non-trivial argument is required for proving refutational completeness. In this paper we describe in detail a decision procedure for the guarded fragment with equality which is based on resolution and superposition. Despite the fact that it applies to a larger fragment, our new procedure is simpler than the one in (de Nivelle 1998) in that we employ a liftable ordering (plus selection) so that we are able to re-use standard results about refutational completeness. Our method is also interesting as there are not so many saturation-based decision procedures for fragments with equality described in the literature. Notable exceptions include (Fermüller & Salzer 1993), where a resolution decision procedure is given for the Ackermann class with equality, and (Bachmair, Ganzinger & Waldmann 1993), where it is shown that a certain superposition strategy decides the monadic class with equality. Nieuwenhuis (1996) proves the decidability of certain shallow equational theories by basic paramodulation.

The advantage of resolution or superposition decision procedures over theoretical procedures based on collapsing models is that the former use syntactic, unification-based inferences to enumerate candidate witnesses of inconsistency. There is experimental evidence (Hustadt & Schmidt 1997) that such procedures perform well in practice, in particular they often will not exhibit the usually exponential or double-exponential worst-case complexity of the respective fragments. Also, when having a flexible saturation theorem prover at hand, such as SPASS (Weidenbach 1997), it suffices to appropriately adjust its parameters in order to efficiently implement the procedure.

The results of this paper can be summarized as follows. (i) Ordered paramodulation with selection is a decision procedure for the GF with equality. No sophisticated redundancy elimination methods are re-

[*]Work supported in part by the ESPRIT Basic Research Working Group 22457 (CCL II).

[†]Work done at ILLC, U. Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam

quired, and a straightforward (liftable) ordering and selection strategy suffice. (ii) The procedure decides the class of guarded clauses which is a proper superclass of the GF with equality. (iii) The worst-case time complexity of the decision procedure is doubly-exponential which is optimal, given that the logic is 2EXPTIME-complete (Grädel 1997). (iv) Guarded clauses with deep terms, although decidable in the case without equality, become undecidable in the equational case. (v) The superposition-based decision method can be extended to the loosely guarded fragment with equality, but is much more involved there. For the extension, hyper-inferences which simultaneously resolve a set of guards are needed. Some non-trivial results are required about the existence of suitable partial inferences to avoid the generation of clauses which are not loosely guarded, together with meta-theorems about the refutational completeness of these partial inferences.

2 The Guarded Fragment

Definition 2.1 The formulas of the *guarded fragment* GF of *function-free* first order logic are inductively defined as follows:

1. $\top$ and $\bot$ are in GF.
2. If A is an atom then A is in GF.
3. GF is cosed under boolean combinations.
4. If $F \in \mathsf{GF}$ and G is an atom, for which every free variable of F is among the arguments of G, then $\forall\overline{x}(G \to F) \in \mathsf{GF}$ (and, equivalently, $\forall\overline{x}(\neg G \vee F) \in \mathsf{GF}$) and $\exists\overline{x}(G \wedge F) \in \mathsf{GF}$, for every sequence of variables $\overline{x}$.

The atoms G which appear as constraints for quantified variables are called *guards*. Equations can also be used as guards. These are examples of guarded formulae:

$$\forall x\,(x \approx x \to p(x)),\quad \exists x\,(p(x) \wedge q(x))$$
$$\forall yz\,(r(y,y,z) \to \bot),\quad \forall xy\,(r(x,y) \to r(y,x))$$
$$\forall xy\,(r(x,y) \to \exists z\; r(y,z))$$
$$\exists x\,[R(w,x) \wedge \forall y\,(R(x,y) \to p(y)\;) \wedge q(x)]$$

The last formula is the translation of the modal formula $\Diamond(\Box p \wedge q)$ with respect to a world w. These are formulae which are not guarded:

$$\forall xy\, p(x,y)$$
$$\forall x\,(\forall y\,(r(x,y) \to r(y,x)))$$
$$\forall x_1x_2x_3\,[p(x_1,x_2) \to p(x_2,x_3) \to p(x_1,x_3)].$$

The second formula illustrates why, in the guarded fragment, a quantification over a sequence of variables $\overline{x}$ cannot generally be replaced by a series of single-variable quantifications without violating the syntactic requirement for the presence of suitable guards. The last formula states the transitivity of p. As this is not guarded, for modal logics such as $S4$ which are based on transitive frames the standard embedding methods lead outside the guarded fragment.

3 The Superposition Calculus

For the decision procedure to be described below we only need a rather weak form of the superposition calculus of Bachmair & Ganzinger (1990), called ordered paramodulation, for which Hsiang & Rusinowitch (1991) have also given a completeness proof. Here (ordered) paramodulation into the larger side of an equation is permitted. We use the symbol $\approx$ to denote formal equality and do not distinguish between equations $s \approx t$ and $t \approx s$. Disequations $\neg(s \approx t)$ will also be written as $s \not\approx t$. The calculus is clausal, where clauses are multisets of literals $L_1, \ldots, L_k$, $k \geq 0$, which we write as disjunctions $L_1 \vee \ldots \vee L_k$. A clause is called *positive* if it does not contain any negative literals. A clause is called *ground* or *propositional* if it contains no variables.

The calculus is parameterized by admissible orderings $\succ$ and selection functions for negative literals Σ, and for each setting of the two parameters it is refutationally complete. For dealing with the orderings it is useful to view non-equational atoms of the form $p(t_1, \ldots, t_k)$, with p a predicate symbol different from equality, as a shorthand notation for an equation $p(t_1, \ldots, t_k) \approx \mathsf{tt}$. In this encoding, the atom is considered a term (in a two-sorted signature with sorts i and o), with tt a distinguished constant of sort o, and where predicates are viewed as functions of sort o, taking arguments of sort i. An *admissible ordering* $\succ$ is any total reduction ordering on ground terms (including non-equational ground atoms) in which tt is minimal. The multiset extension of $\succ$, again denoted $\succ$, is used to compare literals by identifying any positive [negative] equation $s \approx t$ (including the equational encodings of non-equational atoms) with the multiset $\{s,t\}$ $[\{s,t,\mathsf{tt}\}]$. The ordering is extended to non-ground expressions by defining $E \succ E'$ iff, for all ground substitutions σ, $E\sigma \succ E'\sigma$. Although admissible orderings are total on ground terms and literals, they are only partial on non-ground expressions. Whenever a literal L contains a unique maximal term we will denote it by $\max(L)$. A *selection function* Σ selects, in each clause, at most one (occurrence of a) negative literal. This occurrence is called *selected*.

Inferences involve eligible literals. A literal is called *eligible* in a clause C if either it is selected in C (by Σ), or else nothing is selected in C, and it is a maximal literal in C with respect to $\succ$. In particular, a positive literal, since it cannot be selected, is eligible only if the respective clause contains no selected (negative) literal. The inference rules are as follows:

Ordered Factoring. From $A_1 \vee A_2 \vee R$ derive $A_1\sigma \vee R\sigma$ provided A_1 is eligible and σ is the mgu of A_1 and A_2.

Equality Factoring. From $t_1 \approx u \vee t_2 \approx v \vee R$ derive $u\sigma \not\approx v\sigma \vee t_1\sigma \approx v\sigma \vee R\sigma$ provided $t_1 \approx u$ is eligible and σ is the mgu of t_1 and t_2.

Reflexivity Resolution. From $t_1 \not\approx t_2 \vee R$ derive $R\sigma$ provided that $t_1 \not\approx t_2$ is eligible and σ is the mgu of t_1 and t_2.

Resolution. From $A_1 \vee R_1$ and $\neg A_2 \vee R_2$ derive $R_1\sigma \vee R_2\sigma$ provided that both A_1 and $\neg A_2$ are eligible and σ is the mgu of A_1 and A_2.

Ordered Paramodulation. From $t_1 \approx u \vee R_1$ and $L[t_2] \vee R_2$, where t_2 is not a variable, derive $L[u]\sigma \vee R_1\sigma \vee R_2\sigma$ provided that both $t_1 \approx u$ and the literal $L[t_2]$ are eligible, σ is the mgu of t_1 and t_2, and $u \not\succeq t_1$.

The way in which the order restrictions are applied here is *a priori*, i.e. before the unifier is computed. Superposition is complete also if the order restrictions are checked after the substitution is applied to the premises (*a posteriori* checking), or even if they are attached to the clauses and inherited throughout inferences. A priori checking has the advantage that the eligible literals in a clause can be precomputed, before any inference is attempted. On the other hand, a posteriori application is generally more restrictive. For obtaining the theoretical results in the present paper a priori ordering constraints turn out to be sufficiently powerful.

The calculus is refutationally complete for any choice of admissible ordering and selection function. Moreover, the calculus is compatible with a rather powerful notion of redundancy by which don't-care nondeterministic simplification and redundancy elimination can be justified. In particular, tautologies can be eliminated and multiple occurrences of literals in clauses can be deleted. The notion of redundancy allows for much more sophisticated simplification methods which, however, will not be required here, although for good practical performance some of them are indisensable. The fact that non-naive implementations of superposition, such as in the SPASS system, spend most of their execution time on simplification rather than search is what makes them useful in the end.

Theorem 3.1 (Bachmair & Ganzinger, 1990) Let N be a set of clauses that is saturated up to redundancy with respect to the above derivation rules. Then N is unsatisfiable if and only if N contains the empty clause.

4 The Decision Procedure

We will now describe the decision procedure. We define a notion of guarded clauses, and show that guarded formulae can be translated into guarded clause sets. We will obtain a resolution decision procedure by defining a reduction order $\succ$ and a selection function Σ that force an upper bound on the complexity of the derivable clauses.

4.1 Clausal Normal Form Translation

We rely on a specific clausal normalform transformation for the guarded fragment. We may assume that the given formula is in negation normal form, that is, negation is only applied to atoms. We also assume that implications and equivalences have been eliminated by replacing them by equivalent formulas involving conjunction, disjunction, and negation. These standard transformations do not take a formula outside the guarded fragment.

The next step is to replace certain subformulae by fresh names, together with a definition of the name.[1] We abstract universally quantified subformulae to reduce the number of quantifier alternations. Let $\mathcal{F} = \{F_1, \ldots, F_n\}$ be a set of formulae in negation normal form. The *structural transformation* of $\mathcal{GF}$ is obtained by iterating the following transformations: If F is a formula in $\mathcal{F}$ containing a proper subformula of the form $\forall\overline{x}(\neg G \vee H)$, with G the guard, then (i) add a *definition* $\forall\overline{xy}(\neg G \vee \neg\,\alpha(\overline{y}) \vee H)$ to $\mathcal{F}$, and (ii) replace the indicated subformula in F by $\alpha(\overline{y})$. Hereby it is assumed that $\overline{y}$ is the set of variables that occur in G, but not in $\overline{x}$, and that α is a new predicate name that does not occur in $\mathcal{F}$. Observe that the structural transformation, when applied to a set of guarded formulas also yields a set of guarded formulas as result. Moreover, all remaining universal quantifiers are outermost, so that any inner existential quantifier occurs in the scope of all universally quantified variables.

For the purposes of this paper, a straightforward skolemization technique suffices which replaces any applied occurrence of an existentially quantified variable y by a term $f(x_1, \ldots, x_n)$, with f a new *Skolem* function symbol, if $x_1, \ldots, x_n$ are the universally quantified variables, in the scope of which y occurs. After that replacement, all existential quantifiers have

[1]Such transformations are called structural and are, for instance, studied in (Baaz, Fermüller & Leitsch 1994). They are called structural since more of the structure of a formula will be preserved when the formula is factored.

been removed. Finally, to obtain a set of clauses, distribute disjunctions over conjunctions, omit the universal quantifiers (which are all outermost) and consider any conjunction of disjunctions as a set (of clauses).

Example 4.1 Consider the guarded formula
$\exists x\ (n(x) \wedge \forall y\ [\neg(a(x,y) \vee$
$\forall z\ \{\neg p(x,z) \vee \exists x\ (a(x,z) \wedge (\neg b(z,z) \vee \neg c(x,x)))\}])$.
The structural transformation gives the set of formulas
$\exists x\ [n(x) \wedge \alpha(x)]$,
$\forall x,y\ [\neg a(x,y) \vee \neg\alpha(x) \vee \beta(x)]$,
$\forall x,z\ [\neg p(x,z) \vee \neg\beta(x) \vee$
$\exists x\ (a(x,z) \wedge (\neg b(z,z) \vee \neg c(x,x)))]$.
Skolemization yields
$n(c) \wedge \alpha(c)$,
$\forall x,y\ [\neg a(x,y) \vee \neg\alpha(x) \vee \beta(x)]$,
$\forall x,z\ [\neg p(x,z) \vee \neg\beta(x) \vee$
$(a(fxz,z) \wedge (\neg b(z,z) \vee \neg c(fxz,fxz)))]$.
Clausification, finally, produces this set of clauses:
$n(c)$
$\alpha(c)$
$\neg a(x,y) \vee \neg\alpha(x) \vee \beta(x)$
$\neg p(x,z) \vee \neg\beta(x) \vee a(f(x,z),z)$
$\neg p(x,z) \vee \neg\beta(x) \vee \neg b(z,z) \vee \neg c(fxz,fxz)$.

4.2 Guarded Clauses

The result of the tranformation are sets of guarded clauses which, in particular, consist of a specific kind of literals. A literal L is called *simple* if each term in L either is a variable or else a functional term $f(u_1,\ldots,u_m)$, $m \geq 0$, in which all u_j are variables or constants. Hence $p(x,c,f(x))$ and $f(x,c) \not\approx y$ are simple while $\neg p(s(f(0),x))$ and $f(x,s(x)) \approx g(x)$ are not. A clause is called *simple* if all literals are simple. A literal is called *covering* if any non-ground and non-variable subterm in the literal contains all the variables of the literal. An expression is called *functional* if it contains a constant or function symbol, and *non-functional*, otherwise.

Definition 4.2 A simple clause C is called *guarded* if it satisfies the following conditions:
(i) C is a positive, non-functional, single-variable clause; or
(ii) every functional subterm in C contains all the variables of C, and, if C is non-ground, C contains a non-functional negative literal, called a *guard*, which contains all the variables of C.
Clauses of the form (ii) are called *properly guarded*, while the concept of guards is void for the other types of guarded clauses. A set of clauses is called *guarded* if all its clauses are guarded.

Note that if a guarded clause contains a constant it must be a ground clause in which terms have at most depth 2. Also, any literal in a guarded clause is covering. Definition 4.2 is more restrictive than the corresponding definition in (de Nivelle 1998). In the Section 5 we will discuss this issue in more detail.

These are some examples of guarded clauses where suitable guards have been underlined.
$p(0,s(0)) \vee c \not\approx d \vee q(s(0),f(0,0))$
$p(x,x) \vee q(x)$
$\neg p(y,x) \vee \underline{\neg q(x,y,y)} \vee r(x+y,x-y,x)$
$\underline{\neg p(y,x)} \vee \neg q(x,y,y)$
$\underline{x \not\approx y} \vee x \approx (x+y)$
The following clauses are not guarded:
$\neg e(x) \vee e(s(s(x)))$ (not simple)
$\neg p(x) \vee \neg q(y) \vee r(x,y)$ (no guard)
$\neg p(f(x,y)) \vee p(x,y)$ (no guard)
$\neg p(x,y) \vee p(f(x),y)$ (not covering)
$\neg p(x,y) \vee p(0,g(x,y))$ (constant, but non-ground)

Theorem 4.3 The number of different (up to variable renaming) guarded clauses (without duplicate occurrences of literals) over a finite signature has a double exponential upper bound.

Proof. Let a finite signature be given. Define the following parameters:

a_1	the maximal arity of function symbols
a_2	the maximal arity of predicate symbols
a	the maximum of a_1 and a_2
n_1	the number constant and function symbols $+\ a_2$
n_2	the number of predicate symbols
n	the maximum of n_1 and n_2.

The maximal size s of a simple atom is a^2+a+1. Therefore, the number of simple atoms over $\mathcal{S}$ is bound by

$$n^s = n^{a^2+a+1}.$$

Then the number of simple literals (modulo variable renaming) is at most

$$l = 2n^{a^2+a+1}.$$

This is also an upper bound for the maximal number of literals in a clause, since a clause contains at most all possible literals over at most a_2 variables. Then the number of guarded clauses that can be constructed from non-repeated literals is bound by

$$c = 2^l = 2^{2n^{a^2+a+1}}.$$

□

4.3 Preservation of Guardedness

We now show that guarded clauses are closed under the paramodulation inferences so that, using the Theorem 4.3, saturating a given set of clauses under these inferences, combined with eager elimination of duplicate literals in clauses, yields a decision procedure for satisfiability. To that end we need to define an appropriate ordering and selection function. For the ordering $\succ$ we may use any lexicographic path ordering on terms and non-equational atoms based on a precedence $\succ$ such that $f \succ c \succ p \succ \mathsf{tt}$ for any non-constant function symbol f, constant c, and predicate symbol p, respectively. For the selection function Σ we assume that (i) if a clause is non-functional and contains a guard then one of its guards is selected by Σ; (ii) if a clause contains a functional negative literal, one of these is selected; and (iii) if a clause contains a positive functional literal, but no negative functional literal, no literal is selected, so that the maximality principle applies for a literal to be eligible for an inference.

Lemma 4.4 Let L_1, L_2 be two literals of a guarded clause. Assume that L_2 contains a non-ground functional term, while L_1 does not. Then $L_2 \succ L_1$.

Proof. First observe that with the given assumptions the calsue does not contain any constants. Let L_1 be a literal, and let t be a non-ground functional term in L_2. First suppose that L_1 is a non-equational literal of the form $[\neg]p(u_1, \ldots, u_n)$ with variables u_i. Moreover, any of the u_i also occurs in t. With regard to the ordering, non-equational literals of this form are identified with equations $[\neg](p(u_1, \ldots, u_n) \approx \mathsf{tt})$. Let f be the leading function symbol in t. Then f has a precedence greater that any of the symbols in L_2, and as t contains all variables of L_1, we conclude that $p(u_1, \ldots, u_n) \prec t \preceq \max(L_2)$ which implies that $L_1 \prec L_2$.

If L_1 is an equational atom $u \approx v$, by a similar reasoning we infer that $t \succ u$ and $t \succ v$, from which again $L_1 \prec L_2$ is inferred. □

Lemma 4.5 With $\succ$ and Σ as defined above, a literal in a clause is eligible for an inference only if it contains all the variables of the clause.

Lemma 4.6 Let σ be the most general unifier of two simple non-equational atoms $p(t_1, \ldots, t_n)$ and $p(u_1, \ldots, u_n)$. Then $p(t_1, \ldots, t_n)\sigma$ is also simple.

Lemma 4.7 Let A and B be simple atoms such that (i) every variable occurring in B also occurs in A; (ii) every variable that occurs in a functional term of B also occurs in a functional term of A; and (iii) every functional term of B contains all the variables of A. Then for any substitution σ,
(i) if $A\sigma$ is simple, then $B\sigma$ is simple,
(ii) every variable of $B\sigma$ occurs in $A\sigma$,
(iii) every variable occurring in a functional term of $B\sigma$ occurs in a functional term of $A\sigma$.
(iv) Every functional term of $B\sigma$ contains all the variables of $A\sigma$.

As a consequence of the Lemma 4.4, if a clause is non-ground, any eligible literal either contains a (non-ground) functional term or else there is no functional term in the entire clause. The preceeding lemma can therefore be applied to any eligible literal A and any other literal B in a guarded clause.

Lemma 4.8 A factor of a guarded clause is guarded.

Lemma 4.9 An equality factor of a guarded clause is guarded.

Lemma 4.10 A clause obtained by reflexivity resolution from a guarded clause, is guarded.

Proof. The propositional case the lemma is trivial. For reflexivity resolution to be applicable to a non-propositional clause, the clause must be of the form $D = x \not\approx y \vee C$, with guard $x \not\approx y$ and with C not containing a functional term. Clearly, the resolvent has only simple literals and is either the empty clause or has just one variable. In the latter case the resolvent either has a guard or is a positive clause. □

Lemma 4.11 A resolvent of two guarded clauses is guarded.

Proof. Let $C_1 = A_1 \vee D_1$ and $C_2 = \neg A_2 \vee D_2$ be the clauses resolved upon, with σ the mgu of A_1 and A_2. Then the conclusion is the clause $D = D_1\sigma \vee D_2\sigma$. As both A_1 and A_2 are simple, the literal $A_1\sigma$ is also simple. Applying the Lemma 4.7, part (i), we may infer that all literals in D are simple. If there are functional terms in D then these contain the same set of variables, and all the variables of D, cf. Theorem 4.7, parts (iii) and (iv). In order to show that there is a guard in D when one is needed, we distinguish as to whether or not the clauses are ground.

Suppose that one of the C_i is ground. In that case D is ground since literals which are eligible for an inference contain all the variables of a clause.

Let us now assume that both C_1 and C_2 are non-ground. Suppose that C_1 is not a positive clause over one variable. Then C_1 must have a guard $\neg G$, and $\neg G\sigma$

occurs in D. Moreover, A_1 must have a functional term containing all the variables of C_1. (Otherwise $\neg G$ or some other guard of C_1 would be selected and the inference would not be possible). As $A_1\sigma$ is simple, σ assigns a variable to each variable in C_1. Therefore, the literal $\neg G\sigma$ has only variables as arguments. Since $\neg G\sigma$ contains all the variables of $A_1\sigma$, it contains all variables of D, and, hence, is a guard. If C_1 is a positive, single-variable clause, and if $\neg A_2$ is not a guard then any guard of C_2 becomes a guard of D. If there is no guard in D then the resolvent is a single-variable, positive clause.

Finally, the resolvent does not contain a constant unless one of the premises does. In that case both the premise and the resolvent are ground. □

LEMMA 4.12 Any clause obtained by a superposition inference from two guarded clauses is guarded.

Proof. Let $C_1 = L[u] \vee D_1$ be the main premise, $C_2 = t_1 \approx t_2 \vee D_2$ the side premise, and $D = L[t_2]\sigma \vee D_1\sigma \vee D_2\sigma$ be the conclusion, respectively, of the inference, with σ the mgu of t_1 and u.

We first consider the case where C_2 is ground. If t_2 is not a constant then also t_1 is not a constant, as otherwise the ordering constraints would block the inference. Superposition inferences into variables are excluded so that u must be a functional term containing all the variables of the clause. Hence, all variables in u become grounded by σ, D is ground, and contains simple literals only.

If C_2 is non-ground, then $t_1 \approx t_2$ has to contain all the variables and at least one of the t_1 or t_2 is a functional term. (Otherwise the guard in C_2 would be selected and the clause cannot appear as the side premise of the inference.) The ordering restrictions, therefore, imply that t_1 is functional, containing all the variables of the clause, whereas t_2 can be a variable, or a functional term. The possible forms of u are also restricted. u cannot be a variable. u can be a functional term containing all the variables of C_2, or a ground term. Suppose that u is ground and unifiable with t_1. Then u is not a constant, C_2 is ground, and u occurs as an argument to a predicate in C_2. Then, D is a ground clause and is simple since $t_2\sigma$ is either a constant of a functional term with constant arguments. If u is not ground σ is a variable renaming and, in particular, both $D_1\sigma$ and $D_2\sigma$ are guarded. Moreover, $L[t_2]\sigma$ is simple. It is easily checked that the guards of $C_1\sigma$ and $C_2\sigma$ can both serve as guards of D. □

THEOREM 4.13 Let Σ and $\succ$ be as specified. For all the inferences of the ordered paramodulation calculus, if the premises are guarded, so is the conclusion.

THEOREM 4.14 The fragment of guarded clauses is decidable by ordered paramodulation.

Proof. By the Theorem 4.13 all derivable clauses are guarded, and the number of such clauses is finite, cf. Theorem 4.3. In case an ordering constraint for an inference is not obviously satisfied[2] one may simply check whether or not the conclusion is a guarded clause and disregard the inference if it is not. As inferences are sound also in cases where the ordering constraints are violated and as sound consequences may always be added without endangering completeness of the calculus,[3] the theorem follows. □

The theorem can also be extended to guarded clauses combined with unrestricted ground clauses. There one replaces in the initial clause set any ground (sub-) term s of depth greater than 2 by a new constant a_s, together with the defining equation $a_s \approx s$. This preserves satisfiablity and produces a clause set which is guarded.

4.4 Complexity

The complexity of our decision procedure is double exponentional. Grädel has shown in (Grädel 1997) that the decision problem for the guarded fragment with equality is 2EXPTIME-complete, so our procedure is theoretically optimal. We use the fact, cf. Theorem 4.3, that the number of guarded clauses is doubly exponentially bound and show that the saturation process has no primitive operation that has more than exponential complexity.

THEOREM 4.15 The superposition decision procedure can be implemented in 2EXPTIME.

Proof. We reuse the notation defined in the proof of the Theorem 4.3. It is clear that the space complexity of the procedure is dominated by the space that is needed to store the clauses. Hence, we obtain a space

[2]The criterion provided by the Lemma 4.4 can be easily extended into a complete decision procedure for the ordering constraints as they arise with guarded clauses, but we do not want to prove this claim formally.

[3]In general this is not entirely trivial if, as we do here, admissible simplifications are applied eagely. Note that we have to eliminate repeated occurrences of literals in clauses, which is a simplification covered by the notion of redundancy for clauses and inferences in (Bachmair & Ganzinger 1990). There it is shown that redundancy is preserved under the addition of arbitrary logical consequences as well as under the deletion of redundant clauses.

complexity of $s*l*c$. For the time complexity, observe that suitable abstractions of the ordering and selection contraints for the inferences can be checked in polynomial time, cf. the proof of Theorem 4.14. Then one may show that the time needed to do a subsumption check is in $O(l^3 s)$. In fact, one first matches the guard with at most l literals. After that one has to try to match each of the l remaining literals with one of the l literals of the other clause. This gives a total of l^3 attempted matches. Since each matching can take up to s time, this number has to be multiplied by s. Knowing the time complexity for subsumption for guarded clauses, we can estimate the time complexity of our method as a whole. The algorithm has to try all pairs of literals, and in the case that a resolvent is possible, it has to check that the resolvent is not subsumed by one of the existing clauses. This takes time in $O((cl)^2 c(l^3 s))$. This iteration has to be repeated at most c times, resulting in a bound in $O((cl)^2 c^2 l^3 s)$. This number is roughly equal to c^4 which gives the desired double exponential time complexity.

Finally we should also consider the time and space complexity of the clausal normal form translation. It is well-known that the transformation to normal form can take at most single exponential time, which is neglectible compared to the double exponential time obtained above. The (structural) elimination of equivalences is slightly more tricky here as the result has to be a guarded formula. □

5 Weakly Guarded Clauses

The notion of guarded clause as given in the Definition 4.2 is more restrictive than the one given in (de Nivelle 1998). There, terms of arbitrary depth are allowed provided that they are either ground, or contain all variables of the clause. We repeat the formal definition:

Definition 5.1 A clause C is called *weakly guarded*, if (i) every non-ground functional term in C contains all the variables of C; and (ii) if C is non-ground it contains a negative literal, all of which arguments are constants or variables, and which contains all the variables of the clause.

This notion was inspired by the E^+-class. Every clause which is guarded is also weakly guarded, but the converse is not true in general.

Theorem 5.2 Satisfiablity is undecidable for finite sets of weakly guarded clauses if equational atoms are admitted. The fragment remains undecidable if all ground terms are constants.

The Post Correspondence Problem can be reduced to this decision problem. This is essentially due to the fact that equations of the form $f(x,y) \approx x$ make terms containing a variable in a nested functional term equal to a term that violates the condition that functional terms should contain all variables of a clause. For example from the guarded clauses $\neg p(x,y) \vee p(s(f(x,y)))$ and $\neg(x,y) \vee f(x,y) \approx x$ we may deduce the non-guarded clause $\neg p(x,y) \vee p(s(x))\}$. This shows that variables in nested functional terms cannot be combined with equality.

6 The Loosely Guarded Fragment

Our method can be generalized to the so called *loosely guarded fragment*. It obtained by weakening the condition (4) in the Definition 2.1 as follows: If F is loosely guarded and $G_1, \ldots, G_n$ are atoms, with variables as arguments, then the formulae $\forall\overline{x}(G_1 \wedge \cdots \wedge G_n \rightarrow F)$ and $\exists\overline{x}(G_1 \wedge \cdots \wedge G_n \wedge F)$ are loosely guarded, provided that (i) every free variable of F occurs in a G_i, and (ii) every pair of variables y_1, y_2, which are free in F, and of which at least one is among the $\overline{x}$, occur together in one of the G_i. We call the entire conjunction $G_1 \wedge \cdots \wedge G_n$ the *guard* of the formula, and any conjunct a *guard atom*.

In the loosely guarded fragment the until operator can be expressed, which cannot be expressed in the guarded fragment. P until Q can be translated as:

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

Transitivity of R, though, cannot be expressed in the loosely guarded fragment. In the formula

$$\forall x, y, z\; (Rxy \wedge Ryz \rightarrow Rxz)$$

there is no atom in the guard in which the variables x and z co-occur. In fact, Ganzinger, Meyer & Veanes (1999) have shown that allowing for a single transitive relation makes the LGF undecidable in general.

A CNF transformation similar to the one described in the Section 4.1 leads to what we call loosely guarded clauses:

Definition 6.1 A simple clause C is called *loosely guarded* if it satisfies the following conditions:

(i) C is a positive, non-functional, single-variable clause; or

(ii) C contains no constants, every functional subterm in C contains all the variables of C, and C contains a set of negative, non-functional literals $\neg A_1, \ldots, \neg A_n$ in C, $n \geq 0$, called a *guard* of C, such that every pair of variables that occurs in C occurs together in one of the atoms A_i.

Propositional simple clauses are admitted. They have an empty guard.

The main modification of the decision procedure is that in cases where previously a guard atom needed to be selected in a clause now a set of literals may constitute a guard, and some of these have to be resolved simultaneously. Therefore, resolution needs to be generalized to (ordered) hyper-resolution. The basis for this are more general selection functions Σ which now may select an entire, possibly empty set of occurrences of negative literals in a clause. Now a literal is called *selected* if it occurs in the set of selected literals of a clause.

Ordered Hyperresolution with Selection

$$\frac{A_1 \vee R_1 \quad \ldots \quad A_k \vee R_n \quad \neg B_1 \vee \ldots \vee \neg B_n \vee R}{R_1\sigma \vee \ldots \vee R_n\sigma \vee R\sigma}$$

where (i) either the $\neg B_j$ are the literals selected by Σ in the *main premise*, or else $n = 1$, nothing is selected in $\neg B_1 \vee R$, and $\neg B_1$ is maximal in $\neg B_1 \vee R$, (ii) the A_i are eligible in the *side premises* $A_i \vee R_i$, and (iii) σ is the mgu of the tuples $(A_1, \ldots, A_n)$ and $(B_1, \ldots, B_n)$.

Given a hyperresolution inference of this form, we speak of a *partial inference* producing a *partial conclusion* D whenever there exists a non-empty subset $j_1, \ldots, j_k$ of the indices $1 \leq j \leq n$ and

$$D = \bigvee_{1 \leq i \leq k} R_{j_i}\tau \vee \bigvee_{i \notin \{j_1, \ldots, j_k\}} \neg B_i\tau \vee R\tau,$$

with τ the mgu of $(A_{j_1}, \ldots, A_{j_k})$ and $(B_{j_1}, \ldots, B_{j_k})$.

The extended calculus is refutationally complete and compatible with a notion of redundancy by which the usual simplification mechanisms (tautology elimination, condensement, subsumption) can be justified. There is no published result that exactly covers this calculus, but it is easy to generalize the results in (Bachmair & Ganzinger 1990) appropriately.

The orderings which we may use for the decision procedure are the same as for the non-loose case. The selection function Σ should satisfy these restrictions:
(i) If a clause C is non-functional and contains a guard $L_1 \vee \ldots \vee L_k$ then *all* the literals of one of the guards of C are selected by Σ;
(ii) if a clause contains a functional negative literal, *one* of these is selected; and
(iii) if a clause contains a positive functional literal but no negative functional literal, then no literal is selected, so that the maximality principle applies for a literal to be eligible for an inference.

In order to prove that with this ordering and selection strategy, ordered paramodulation becomes a decision procedure for the LGF, two problems have to be solved. There are two more complications in the loosely guarded case. The first problem is that conclusions of inferences might become too deep.

Example 6.2 (de Nivelle & Rijke, 1999) The following clause D is loosely guarded:

$$\neg a_1(x, y) \vee \neg a_2(y, z) \vee \neg a_3(z, x) \vee b_1(x, y) \vee b_2(y, z) \vee b_3(z, x)$$

There are no functional terms, therefore the three guard literals are selected. The following three clauses are candidates for a hyperresolution inference:

$$\begin{aligned} C_1 &= \neg p_1(u) \vee a_1(fu, fu), \\ C_2 &= \neg p_2(v) \vee a_2(v, gv), \\ C_3 &= \neg p_3(w) \vee a_3(gw, w), \end{aligned}$$

It is possible to derive the hyperresolvent

$$\neg p_1(u) \vee \neg p_2(fu) \vee \neg p_3(fu) \vee b_1(fu, fu) \vee b_2(fu, gfu) \vee b_3(gfu, fu),$$

with an mgu $\sigma = [x, y, v, w := fu, z := gfu]$. This resolvent has a term depth greater than 2 which is not admitted for a loosely guarded clause

A remedy to this problem is to only resolve D with a suitable subset of the side premises C_i. In the example, if we only resolve the second and third guard literal of D with C_2 and C_3, respectively, we obtain the partial conclusion

$$\neg a_1(w, w) \vee \neg p_2(w) \vee \neg p_3(w) \vee b_1(w, w) \vee b_2(w, gw) \vee b_3(gw, w).$$

The mgu of the partial inference is $[y, v, x := w, z := gw]$. This clause is loosely guarded, in particular, not too deep. It turns out that if an inference is possible then one of the possible partial conclusions will be a guarded clause. The proof makes use of the subsequent lemma which is a special case of a theorem in (de Nivelle & Rijke 1999).

Lemma 6.3 Let $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$ be $2n \geq 2$ simple literals such that
(i) the B_i are non-functional;
(ii) for all x, y in $\mathrm{Var}(B_1, \ldots, B_n)$ there is an B_i such that x, y is in $\mathrm{Var}(B_i)$;
(iii) the A_j are covering and functional;

(iv) A_i and A_j, for $i \neq j$, have no common variables;
(v) the A_i and the B_j have no common variables;
(vi) the tuples $(A_1, \ldots, A_n)$ and $(B_1, \ldots, B_n)$ are unifiable.

Then there exists a non-empty subset $j_1, \ldots, j_k$ of the indices $1 \leq j \leq n$ such that the tuples $(A_{j_1}, \ldots, A_{j_k})$ and $(B_{j_1}, \ldots, B_{j_k})$ are unifiable with an mgu τ and
(i) any of the $A_{j_i}\tau$ ($= B_{j_i}\tau$) is simple and covering;
(ii) if x is a variable in any of the B_i or A_{j_i} and if y is a variable in $y\tau$ then y also occurs in $A_{j_1}\tau$.

The proof which is given in full detail for the somewhat more general theorem in (de Nivelle & Rijke 1999) is based on this observation: Let us assume, for simplicity, that the A_i are non-ground and that all non-constant symbols are binary. Then any of the A_i is of the form $p(u, fuv)$, $p(fuv, v)$, or $p(fuv, guv)$, with more variants arising from exchanging u and v in one of the arguments. In the binary case, if we neglect the trivial one-variable case, any of the guard atoms is of the form $p(x, y)$, with different variables x and y. The problem of unifying all the A_i with the corresponding B_i, therefore, induces at least one unification problem of the form $x = fxy$, $y = fxy$, $x = y$, $fxy = fxy$, or $fxy = fyx$ on any pair x, y of variables in $\mathrm{Var}(B_1, \ldots, B_n)$. This is a consequence of the co-occurrence requirement (ii). If the unification problem is solvable with an mgu σ then the substitution instances $x\sigma$ of the variables x in $\mathrm{Var}(B_1, \ldots, B_n)$ are totally ordered by the subterm ordering. Picking for the j_i those atoms in which the variables have maximal depth M or depth $M - 1$ solves the problem.

The lemma covers exactly those unification problems which arise from hyperresolution inferences with guard atoms B_i and corresponding positive atoms A_i. For the latter to be eligible for an inference they all have to contain a functional term. In other words, with the class of orderings $\succ$ and selection functions Σ which we consider for the LGF, we obtain this theorem:

Theorem 6.4 Suppose there is an inference by hyperresolution with respect to $\succ$ and Σ. Then one of the paritial inferences produces a (partial) conclusion which is a guarded clause.

Why does the existence of suitable partial inferences solve our problem with hyperresolution?

The answer is that the calculus remains complete if, for any potential hyperinference from side premises $C_1, \ldots, C_k$ and main premise D, rather than adding the full conclusion, we add any don't-care non-deterministically chosen partial conclusion. (A proof of this fact in the non-equational case has been given in (Bachmair & Ganzinger 1997), and the extension to the equational case which we consider here is not difficult.) The criterion for which partial conclusion to choose is simply that the conclusion should be a guarded clause. With this modification of the calculus, the class of guarded clauses is closed under its inferences.

A second, much more simple problem, arises from the fact that loosely guarded clauses over any given finite signature may be arbitrarily long. Fortunately it is not difficult to see that the set of guarded clauses that can be derived with our inference system from an initially given finite set of guarded clauses is finite. This is an immediate consequence of the fact that the number of variables does not increase during an inference:

Lemma 6.5 If D is the [partial] conclusion of an inference from premises C_i then $|\mathrm{Var}(D)| \leq \max(|\mathrm{Var}(C_i)|)$.

Altogether we obtain:

Theorem 6.6 Ordered Paramodulation with hyperresolution based on selection is a decision procedure for the LGF.

7 Conclusions

We have shown that it is possible to effectively decide the [loosely] guarded fragment with equality by superposition-based saturation provers. There is hope that usable decision procedures can be obtained from these results with existing standard theorem provers. This hope is supported by our theoretical optimality result (in the non-loose case) and by experimental evidence that has been obtained in using these theorem proving techniques in related application domains (Hustadt & Schmidt 1997). The GF has turned out to be a fragment of first-order logic with equality for which it is especially easy to configure superposition into an optimal decision procedure. Although the complexity issue has been neglected by and large in the literature on resolution-based decision procedures, we believe that in most cases of fragments which are complete for a particular time complexity class, the resolution-based methods can be implemented it this time bound. (Things are different for space complexity classes such as PSPACE and local theorem proving methods based on resolution and superposition where the reuse of space as is standard with tableau methods is not so straightforward.) The loosely guarded case is more tricky. However this paper also demonstrates that the theory of saturation-based theorem proving is sufficiently developed to be able to solve the problems without having to deal with technically difficult proof-theoretic arguments.

References

Andréka, H., van Benthem, J. & Németi, I. (1996), Modal languages and bounded fragments of predicate logic, Technical report, ILLC.

Baaz, M., Fermüller, C. & Leitsch, A. (1994), A non-elementary speed up in proof length by structural clause form transformation, *in* 'Proc. LICS'94'.

Bachmair, L. & Ganzinger, H. (1990), On restrictions of ordered paramodulation with simplification, *in* M. Stickel, ed., 'Proc. 10th Int. Conf. on Automated Deduction, Kaiserslautern', Vol. 449 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 427–441.

Bachmair, L. & Ganzinger, H. (1997), A theory of resolution, Research Report MPI-I-97-2-005, Max-Planck-Institut für Informatik, Saarbrücken, Saarbrücken. To appear in the Handbook of Automated Reasoning.

Bachmair, L., Ganzinger, H. & Waldmann, U. (1993), Superposition with simplification as a decision procedure for the monadic class with equality, *in* G. Gottlob, A. Leitsch & D. Mundici, eds, 'Proc. of Third Kurt Gödel Colloquium, KGC'93', Vol. 713 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 83–96. Revised version of Technical Report MPI-I-93-204.

de Nivelle, H. (1998), A resolution decision procedure for the guarded fragment, *in* C. Kirchner & H. Kirchner, eds, 'CADE-15', pp. 191–204.

de Nivelle, H. & Rijke, M. (1999), A resolution decision procedure for the guarded fragment, Manuscript.

Fermüller, C. G. & Salzer, G. (1993), Ordered paramodulation and resolution as decision procedure, *in* A. Voronkov, ed., 'Proceedings of the 4th International Conference on Logic Programming and Automated Reasoning (LPAR'93)', Vol. 698 of *LNAI*, Springer Verlag, St. Petersburg, Russia, pp. 122–133.

Ganzinger, H., Meyer, C. & Veanes, M. (1999), The two-variable guarded fragment with transitive relations, *in* 'Proc. 14th IEEE Symposium on Logic in Computer Science', IEEE Computer Society Press, pp. ??–?? To appear.

Grädel, E. (1997), On the restraining power of guards, Manuscript.

Hsiang, J. & Rusinowitch, M. (1991), 'Proving refutational completeness of theorem proving strategies: The transfinite semantic tree method', *J. Association for Computing Machinery* **38**(3), 559–587.

Hustadt, U. & Schmidt, R. A. (1997), On evaluating decision procedures for modal logics, *in* M. E. Pollack, ed., 'Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)', International Joint Conferences on Artificial Intelligence, Inc. (IJCAII) and Japanese Society for Artificial Intelligence (JSAI), Morgan Kaufmann, Nagoya, Japan, pp. 202–207.

Nieuwenhuis, R. (1996), Basic paramodulation and decidable theories, *in* 'Proceedings of the Eleventh Annual IEEE Symposium On Logic In Computer Science (LICS'96)', IEEE Computer Society Press, pp. 473–483.

Weidenbach, C. (1997), 'Spass version 0.49', *J. Automated Reasoning* **18**(2), 247–252.