Combining Resolution Decision Procedures

Yevgeny Kazakov

MPI für Informatik, D-66123 Saarbrücken, Germany
`ykazakov@mpi-sb.mpg.de`

Abstract. We present resolution-based decision procedures for the guarded, two-variable and monadic fragments without equality in a uniform way and show how they can be combined. In this way, new decidable fragments are obtained. We make use of a novel technique for describing resolution decision procedures by means of clause schemes. The scheme notation provides a convenient way of specifying decision procedures, proving their correctness and analyzing complexity of associated decision problems. We also show that many decidability results obtained in the paper cannot be extended to the case with equality.

Keywords: automated deduction, decision procedures, combination.

1 Introduction and Motivation

First-order logic has always been *the* language for specifications of problems in mathematics and computer science. Since Church and Turing in 1930's proved that the *classical decision problem*: "Given a first-order sentence, decide its validity", is algorithmically unsolvable, logicians shifted their effort in trying to identify *decidable fragments* of first-order logic. Many decidable fragments of first-order logic have been found by imposing certain syntactical restrictions on quantifier pattern, number of variables, or arities of predicate and functional symbols that can be used (for an overview of these and related results see Börger, Grädel & Gurevich 1997). Among them, *monadic*, *two-variable* and *guarded fragments* have attracted considerable attention due to potential applications in artificial intelligence and program analysis (Grädel & Otto 1999, Grädel 1999, Lutz, Sattler & Wolter 2001, Bachmair, Ganzinger & Waldmann 1993). The restrictions yielding decidability of these fragments are of quite orthogonal nature: the two-variable fragment allows only for formulas that are constructed using at most two variable names; in the guarded fragment every quantification should have the bounded form: $\forall\overline{x}.(G \rightarrow F)$ or $\exists\overline{x}.(G \wedge F)$, where G is an atom-"guard", containing all free variables of F; in the monadic fragment formulas can be constructed without restrictions but from unary predicate symbols only.

As the number of applications for first-order theorem proving grows, more decidable fragments with broad expressive power and clear complexity are required for developing practical decision algorithms. One way of providing new decidable fragments could be in combining the known decidable fragments. For example, if the axioms of a theory fall into different decidable fragments, it is appropriate to consider a new fragment obtained by taking *conjunctions* of formulas in the old fragments. In this paper we investigate decidability and complexity for combinations of guarded, two-variable and monadic fragments that we call a *recursive composition*. The idea of the combination is to allow exchanging the formulae between fragments as *new atoms*. For example, the formula:

$$\forall x.[\texttt{Nat}(x) \rightarrow \texttt{Nat}(s(x))] \wedge \forall xy.[\texttt{Nat}(x) \wedge \texttt{Nat}(y) \rightarrow \overbrace{\exists z.(\texttt{Sum}(x,y,z) \wedge \texttt{Nat}(z))}^{\texttt{Summable}(x,y)}]$$

that represents a partial specification for natural numbers, is a conjunct of the monadic formula and the formula having a subformula from the guarded fragment. If we treat this subformula as a new atom $\texttt{Summable}(x, y)$, then the second conjunct becomes a two-variable formula. Thus, the whole formula belongs in some sense to a combination of the *full* monadic fragment, two-variable fragment and the guarded fragment, but to none of the components. We show that recursive composition of any of these fragments yields a decidable fragment of first-order logic as long as the equality is not allowed.

We obtain decidability and sharp complexity results for the fragments and their combinations using the *ordered resolution calculus*. Saturation strategies based on ordered resolution have been successfully applied for obtaining decision procedures for the monadic fragment (Joyner Jr. 1976), the two-variable fragment (de Nivelle 2000) and the guarded fragment (de Nivelle & de Rijke 2003). Our combinational approach is closely related to (Armando, Ranise & Rusinowitch 2001), where the *superposition calculus* was used for deciding existential fragments of certain equational theories and their combinations. A decision procedure for a combination has been obtained there by inspecting all possible cross-inferences between clause classes for each individual theory.

The contributions of the paper can be summarized as follows. First, we present resolution decision procedures for the guarded, two-variable and monadic fragments without equality in a *uniform way*, as a sequence of several basic operations. Second, we obtain new theoretical results about combination of these fragments: ***(i)*** the recursive composition of any of these three fragments is a decidable fragment and ***(ii)*** the complexity of the satisfiability problem for combined fragments is the "maximal complexity" of the fragments it is composed from. And finally, we show that many of these decidability results cannot be carried out to the case with equality. For our decision procedures we make use of a special *scheme* notation for clauses, that allows to express clause classes and its saturations in a very concise form. This may provide a formal foundation for specifying decision procedures and proving their correctness.

2 Preliminaries

We shall use a standard notation for first-order logic. An *expression* is either a term or a literal. A *literal symbol* l is either a or $\neg a$, where a is a predicate symbol. An *expression symbol* e is either a functional symbol f or a literal symbol l. We write literals and expressions using literal symbols and expression symbols as follows: $L = l(t_1,...,t_n)$, $E = e(t_1,...,t_n)$. As usual, a clause is a disjunction of literals $C = L_1 \vee \cdots \vee L_n$. We use the shortcuts ⩙ for conjunction or disjunction and $Q\overline{x}.F$ for either $\exists\overline{x}.F$ or $\forall\overline{x}.F$, where $\overline{x}$ is some vector of variables. $F(\overline{x})$ denotes a formula whose free variables are among $\overline{x}$: $\mathit{free}[F(\overline{x})] \subseteq \overline{x}$. $F[t]$ is a formula with an indicated occurrence of some subterm t and $F[s/t]$ is the result of the replacement of this occurrence by the term s. We denote by $dp(E)$ the term depth of the expression E. The *width* $wd(F)$ of a *formula* F is the maximal number of free variables in subformulas of F. The *width of a clause* C is the number of variables in C.

2.1 The Framework of Resolution Theorem Proving

For describing the decision procedures we use the well-known ordered resolution calculus with selection $\mathcal{OR}^{\succ}_{Sel}$ enhanced with additional simplification rules. Our presentation of the calculus is very close to (Bachmair & Ganzinger 2001). The ordered resolution calculus $\mathcal{OR}^{\succ}_{Sel}$ is parametrized by an admissible ordering $\succ$ and a selection function *Sel*. A partial ordering $\succ$ on atoms is *admissible* (for $\mathcal{OR}^{\succ}_{Sel}$) if ***(i)*** $\succ$ is *liftable*: $A_1 \succ A_2$ implies $A_1\sigma \succ A_2\sigma$ for any substitution σ and ***(ii)*** $\succ$ is a total reduction ordering on ground atoms. Although resolution remains complete for a much wider class of orderings, admissible orderings are better understood and widely used in existing theorem provers. The examples of admissible orderings are the *lexicographic path ordering LPO* and the *Knuth-Bendix ordering KBO*.

The ordering $\succ$ is extended on literals by comparing $L = A$ as the multiset $\{A\}$ and $L = \neg A$ as the multiset $\{A, A\}$. The ordering on clauses is the multiset extension of the ordering on literals. Given a clause C, we say that a literal $L \in C$, is *maximal in* C if there is no L' in C, with $L' \succ L$. A *selection function Sel* assigns a set of negative literal to every clause, which we call *selected literals*. A literal L is *eligible* in a clause C if it is either selected: $L \in Sel(C)$, or otherwise nothing is selected and L is maximal in C.

The ordered resolution calculus $\mathcal{OR}^{\succ}_{Sel}$ consists of two inference rules below. We mark eligible literals with "star" and underline the expressions to be unified:

Ordered Resolution

$$OR\colon \frac{C \vee \underline{A}^* \quad D \vee \neg\underline{B}^*}{C\sigma \vee D\sigma}\ \Bigg|\ \textit{where (i) } \sigma = mgu(A,B),\ \textit{(ii) } A \textit{ and } \neg B \textit{ are eligible.}$$

Ordered Factoring

$$OF\colon \frac{C \vee \underline{A}^* \vee \underline{B}}{C\sigma \vee A\sigma}\ \Bigg|\ \textit{where (i) } \sigma = mgu(A,B),\ \textit{(ii) } A \textit{ is eligible.}$$

The calculus $\mathcal{OR}^{\succ}_{Sel}$ is refutationally complete for any choice of an admissible ordering $\succ$ and a selection function *Sel*. Moreover, the calculus is compatible with a general notion of redundancy which allows to make use of additional simplification rules. A clause C is called *redundant* w.r.t. a clause set N if every ground instance of C follows from *smaller* ground instances of clauses from N. A clause set N is *saturated up to redundancy* if the conclusion of every inference from N is either contained in N or else is redundant w.r.t. N.

Theorem 1. (Bachmair & Ganzinger 2001) *Let N be a clause set that is saturated up to redundancy in $\mathcal{OR}^{\succ}_{Sel}$. Then N is satisfiable iff N does not contain the empty clause.*

For our decision procedures we do not need the full power of redundancy but rather additional simplification rules. A (non-deterministic) inference rule $S \vdash S_1 \parallel S_2 \cdots \parallel S_k$ producing one of the clause sets S_i from the clause set S is called *sound* if every model of S can be extended to a model for some S_i with $1 \leq i \leq k$. Additionally, if *every* set S_i makes some clause from S redundant, the rule is called a *simplification* rule.

Given a set of clauses N, a theorem prover based on ordered resolution non-deterministically computes a *saturation* of N by adding conclusions of inference rules to N and marking[1] redundant clauses as *deleted* so that they do not participate in further inferences. If the process terminates without deriving the empty clause, then a set of the clauses $\mathcal{OR}^{\succ}_{Sel}(N)$ is computed that is saturated in $\mathcal{OR}^{\succ}_{Sel}$ up to redundancy. Theorem 1 then implies that the clause set N is satisfiable, since only satisfiability preserving transformations $N \Rightarrow \cdots \Rightarrow \mathcal{OR}^{\succ}_{Sel}(N)$ were applied to N. Note that termination of a saturation process is a key issue of using resolution as a decision procedure. If any application of inference rules is *a priori* guaranteed to terminate for a clause set N, then satisfiability of N can be decided in finite time by exploring the *finite* saturation tree for N.

In our paper we use the following simplification rules:

Elimination of Duplicate Literals $$ED\colon \frac{[\![\ C \vee D \vee D\ ]\!]}{C \vee D}$$

Splitting $$SP\colon \frac{[\![\ C \vee D\ ]\!]}{C \parallel D} \ \Bigg|\ \textit{where } C \textit{ and } D \textit{ are nonempty and variable disjoint clauses.}$$

An additional simplification rule will be introduced later, when a certain class of orderings is considered. We indicate redundant premises of rules by enclosing them in double brackets. The simplification rules are applied *eagerly*, that is before any resolution or factoring inference is made. In particular, in the sequel we assume that no clause contain several occurrences of the same literal.

2.2 Schemes of Expressions and Clauses

To describe resolution-based decision procedures we have to reason about sets of clauses. We introduce a special notation using which sets of clauses can be represented in a compact form. We extend our vocabulary with additional symbols called *signature groups* that represent sets of functional symbols: *function groups*, predicate symbols: *predicate groups* or literal symbols: *literal groups*. We allow to use these symbols in expressions as usual functional and literal symbols and to distinguish them, we use small letters with the "hat" $\hat{g}$. For instance, if $\hat{f}_{all}$ denotes the set of all functional symbols, we write $\hat{f}_{all}(t)$ meaning a term of the form $f(t)$ where $f \in \hat{f}_{all}$ (the formal definition follows below). We adopt the following notation for referring to arguments of expressions. By writing $e\langle !t_1, ..., !t_n, s_1, ..., s_m\rangle$ we mean an expression starting with the expression symbol e, having all arguments $t_1, ..., t_n$ and optional arguments $s_1, ..., s_m$

[1] Clauses are not removed from the set to avoid repetition of generation/deletion of the same redundant clauses.

(arranged in some way). Formally, the set of *term schemes*, *literal schemes* and *clause schemes* are defined respectively as follows:

$$\begin{aligned}\hat{T}m &::= \; x \mid \hat{f}(\hat{t}_1,...,\hat{t}_n) \mid \hat{f}\langle !\hat{t}_1,...,!\hat{t}_n,\hat{s}_1,...,\hat{s}_m\rangle,\;\; n\geq 0, m\geq 0.\\ \hat{L}t &::= \qquad \hat{l}(\hat{t}_1,...,\hat{t}_n) \mid \hat{l}\langle !\hat{t}_1,...,!\hat{t}_n,\hat{s}_1,...,\hat{s}_m\rangle,\;\; n\geq 0, m\geq 0.\\ \hat{C}l &::= \hat{L} \mid !\hat{L} \mid \hat{C}_1 \vee \hat{C}_2.\end{aligned}$$

where $\hat{f}$ is a functional group, $\hat{l}$ is a literal group, $\hat{t}_i$, $\hat{s}_j$ with $1\leq i\leq n, 1\leq j\leq m$ are term schemes, $\hat{L}$ is a literal scheme and $\hat{C}_1,\hat{C}_2$ are clause schemes. For convenience, we assume that every functional and literal symbol acts as a singleton group consisting of itself, so usual terms and clauses are term schemes and clause schemes as well.

Each term scheme $\hat{t}$, literal scheme $\hat{L}$ and clause scheme $\hat{C}$ represents a set $\langle\hat{t}\rangle$, $\langle\hat{L}\rangle$ and $\langle\hat{C}\rangle$ of terms, literals and clauses respectively, as defined below:

$$\begin{aligned}\langle\hat{T}m\rangle,\langle\hat{L}t\rangle : = \qquad & \langle x\rangle\!:\{x\} \qquad \mid\\ \langle\hat{g}(\hat{t}_1,...,\hat{t}_n)\rangle\!:& \{g(t_1,...,t_n) \mid g\in\hat{g},\; t_i\in\langle\hat{t}_i\rangle, 1\leq i\leq n\} \qquad \mid\\ \langle\hat{g}\langle !\hat{t}_1,...,!\hat{t}_n,\hat{s}_1,...,\hat{s}_m\rangle\rangle\!:& \{g(h_1,...,h_k) \mid g\in\hat{g},\; \{h_1,...,h_k\}\cap\langle\hat{t}_i\rangle\neq\emptyset, 1\leq i\leq n,\\ & \{h_1,...,h_k\}\subseteq \cup_{i=1}^{n}\langle\hat{t}_i\rangle \cup_{j=1}^{m}\langle\hat{s}_j\rangle\}.\end{aligned}$$

$$\begin{aligned}\langle\hat{C}l\rangle = \qquad \langle\hat{L}\rangle\!:& \{L_1\vee\cdots\vee L_k \mid \boldsymbol{k\geq 0},\; L_i\in\langle\hat{L}\rangle,\; 1\leq i\leq k\} \qquad \mid\\ \langle !\hat{L}\rangle\!:& \{L_1\vee\cdots\vee L_k \mid \boldsymbol{k\geq 1},\; L_i\in\langle\hat{L}\rangle,\; 1\leq i\leq k\} \qquad \mid\\ \langle\hat{C}_1\vee\hat{C}_2\rangle\!:& \{C_1\vee C_2 \mid C_1\in\langle\hat{C}_1\rangle,\; C_2\in\langle\hat{C}_2\rangle\}.\end{aligned}$$

We use the shortcuts $\hat{e}(.,\overline{x},.)$, $\hat{e}\langle .,\overline{x},.\rangle$ and $\hat{e}\langle .,!\overline{x},.\rangle$ where $\overline{x}$ is a vector $x_1,...,x_n$, to stand for $\hat{e}(.,x_1,...,x_n,.)$, $\hat{e}\langle .,x_1,...,x_n,.\rangle$ and $\hat{e}\langle .,!x_1,...,!x_n,.\rangle$ respectively. We write $..\vee\neg!\hat{A}\vee..$ in clause schemes instead of $..\vee !\neg\hat{A}\vee..$, where $\hat{A}$ is either of the form $\hat{a}(...)$ or $\hat{a}\langle...\rangle$. In fact we use variable vectors $\overline{x}$ and functional symbols without "hat" f as *parameters of clause schemes.* A clause scheme $\hat{C}(\overline{x},f,...)$ with parameters $\overline{x},f$,... represents the union $\langle\hat{C}\rangle := \cup_\eta\langle C\eta\rangle$ for all substitutions η of vectors $x_1,...,x_n$ for $\overline{x}$, function symbols for f, etc..

Example 1. Suppose $\hat{a}$ is a predicate group consisting of all predicate symbols and $\hat{\alpha} := \{\hat{a},\neg\hat{a}\}$ is a literal group consisting of all literal symbols. Then the clause scheme $\hat{C} = \neg!\hat{a}\langle !\overline{x}\rangle \vee \hat{\alpha}\langle !f(\overline{x}),\overline{x}\rangle$ has two parameters: $\overline{x}$ and f. Any clause $C\in\langle\hat{C}\rangle$ corresponds to some choice of these parameters $\overline{x}\Rightarrow x_1,...,x_n$, $f\Rightarrow f'$. The clause C should have a nonempty subset of negative literals containing all variables $x_1,...,x_n$ and no other arguments. Other literals of C should contain the subterm $f'(x_1,...,x_n)$ as an argument and possibly some variables from $x_1,...,x_n$. In particular, $\langle\hat{C}\rangle$ contains the clauses $\neg a(x,y,x)\vee b(y,f'(x,y))$, $\neg b(x,y)\vee\neg b(y,x)$ and $\neg p\vee\neg q(c,c)$, but not the clauses $\neg a(x,y,x)\vee b(f'(x,y),f'(y,x))$ or $\neg b(y,f'(x,y))$.

3 Resolution-Based Decision Procedures

Decision procedures based on the resolution principle have been around since (Joyner Jr. 1976), who used A-ordering refinements of resolution for deciding some prefix-vocabulary classes. His method has been extended later to capture a variety of other decidable classes. (For an overview of the methods and results along this line see Fermüller, Leitsch, Hustadt & Tammet 2001). Resolution-based decision procedures usually comprise a smart clause normal form transformation with an appropriate choice of an ordering and a selection function. These parameters must be set in such a way that prevents growth of clause depth and width in inferences. Fermüller, Leitsch, Tammet & Zamov (1993) have introduced the notion of covering expressions using which, bounds on clause depth and width can be easily established for different saturation strategies. An expression E is called *covering* if all functional subterms of E contain *all variables* of E. For example, the atom $p(f(x,y),y)$ is covering, whereas, say $p(x,g(y))$ is not. The essential property of covering expressions is that a unification of two covering expression results in a covering expression whose depth is bounded by the maximal depth of the expressions that are unified.

Theorem 2. (Fermüller et al. 1993) *Let E_1 and E_2 be two covering expressions with $dp(E_1) \geq dp(E_2)$ and let $\sigma := mgu(E_1, E_2)$. Then $\sigma : \overline{x} \to \overline{u}$, where $\overline{x} = free(E_1)$ and $\overline{u}$ is some vector of variables.*

3.1 Deciding the Guarded Fragment

Our decision procedure for the guarded fragment is very close to the one given in (de Nivelle & de Rijke 2003, Ganzinger & de Nivelle 1999) however the clause class that we obtain allows to show a sharper complexity bound. The procedure is easier to describe using a recursive definition for the *guarded formulas*:

$$\mathcal{GF} ::= \mathtt{A} \mid \mathtt{F}_1 \vee \mathtt{F}_2 \mid \mathtt{F}_1 \wedge \mathtt{F}_2 \mid \neg\mathtt{F}_1 \mid \forall\overline{x}.(\mathtt{G} \to \mathtt{F}_1) \mid \exists\overline{x}.(\mathtt{G} \wedge \mathtt{F}_1).$$

where $\mathtt{A}$ is an atom, $\mathtt{F}_i, i = 1,2$ are guarded formulas, and $\mathtt{G}$ is an atom called *the guard* containing all free variables of $\mathtt{F}_1$. For example, the seriality axiom $S := \forall x.(\mathtt{V}(x) \to \exists y.[\mathtt{E}(x,y) \wedge \mathtt{V}(y)])$ is a guarded formula, whereas the transitivity axiom: $\forall xyz.(\mathtt{T}(x,y) \wedge \mathtt{T}(y,z) \to \mathtt{T}(x,z))$ is not.

Clause normal form translation. The translation of a guarded formula into a clause normal form (CNF) is done in two steps. First, the formula is transformed into the *negation normal form (NNF)* which falls into the following fragment:

$$[\mathcal{GF}]^{nnf} ::= (\neg)\mathtt{A} \mid \mathtt{F}_1 \vee \mathtt{F}_2 \mid \mathtt{F}_1 \wedge \mathtt{F}_2 \mid \forall\overline{y}.(\mathtt{G} \to \mathtt{F}_1) \mid \exists\overline{y}.(\mathtt{G} \wedge \mathtt{F}_1).$$

we extend this fragment by dropping the restrictions of for existential part:

$$[\mathcal{GF}]^{nnf}_w ::= (\neg)\mathtt{A} \mid \mathtt{F}_1 \vee \mathtt{F}_2 \mid \mathtt{F}_1 \wedge \mathtt{F}_2 \mid \forall\overline{y}.(\mathtt{G} \to \mathtt{F}_1) \mid \exists \boldsymbol{y}.\mathbf{F_1}.$$

After that we apply a so-called *structural transformation* for a sentence $\mathtt{F}$ of this form (we took the existential closure of a formula from $[\mathcal{GF}]^{nnf}_w$) by introducing *definitions* for its subformulas. We assume that to every subformula $\mathtt{F}'$ of $\mathtt{F}$, corresponding to a case in the recursive definition, a unique predicate $P_{\mathtt{F}'} = p_{\mathtt{F}'}(\overline{x})$ is assigned, where $\overline{x} = free[\mathtt{F}']$. The result of the structural transformation is the formula $P_{\mathtt{F}} \wedge [\mathtt{F}]^{st}_g$, where $[\mathtt{F}]^{st}_g$ is recursively defined for $\mathtt{F} \in [\mathcal{GF}]^{nnf}_w$ as follows:

$$\begin{array}{rl|l}
[\mathtt{F}]_g^{st}{:=}[(\neg)\mathtt{A}]_g^{st}{:}\,\forall\overline{x}.(P_{\mathtt{F}}\rightarrow(\neg)\mathtt{A}) & & \neg p_{\mathtt{F}}(\overline{x})\vee(\neg)a\langle\overline{x}\rangle \\
{}[\mathtt{F}_1 \mathbin{\vee\!\!\!\wedge} \mathtt{F}_2]_g^{st}{:}\,\forall\overline{x}.(P_{\mathtt{F}}\rightarrow[P_{\mathtt{F}_1}\mathbin{\vee\!\!\!\wedge}P_{\mathtt{F}_2}])\wedge[\mathtt{F}_1]_g^{st}\wedge[\mathtt{F}_2]_g^{st} & & \neg p_{\mathtt{F}}(\overline{x})\vee p_{\mathtt{F}_i}\langle\overline{x}\rangle\ [\vee\ p_{\mathtt{F}_j}\langle\overline{x}\rangle] \\
{}[\forall\overline{y}.(\mathtt{G}\rightarrow\mathtt{F}_1)]_g^{st}{:}\,\forall\overline{x}.(P_{\mathtt{F}}\rightarrow\forall\overline{y}.[\mathtt{G}\rightarrow P_{\mathtt{F}_1}])\wedge[\mathtt{F}_1]_g^{st} & & \neg g\langle !\overline{x},!\overline{y}\rangle\vee\neg p_{\mathtt{F}}(\overline{x})\vee p_{\mathtt{F}_1}\langle\overline{x},\overline{y}\rangle \\
{}[\exists y.\mathtt{F}_1]_g^{st}{:}\,\forall\overline{x}.(P_{\mathtt{F}}\rightarrow\exists y.P_{\mathtt{F}_1})\wedge[\mathtt{F}_1]_g^{st}. & & \neg p_{\mathtt{F}}(\overline{x})\vee p_{\mathtt{F}_1}\langle f(\overline{x}),!\overline{x}\rangle
\end{array}$$

The transformation unfolds a formula by its definition and replaces its subformulas with fresh predicates. These predicates are defined in separate conjuncts by means of the subformulas that they replace, and the process is iterated recursively. Every transformation step preserves (un)satisfiability of formulas, therefore, $\mathtt{F}$ is satisfiable whenever $P_{\mathtt{F}}\wedge[\mathtt{F}]_g^{st}$ is.

Example 2. The structural transformation for the seriality axiom given above produce the sentence: $p_0\wedge[S]_g^{st}=p_0\wedge[p_0\rightarrow\forall x.(\mathtt{V}(x)\rightarrow p_1(x))]\wedge$
$\wedge\forall x.[p_1(x)\rightarrow\exists y.(p_2(x,y)\wedge p_3(y))]\wedge\forall xy.[p_2(x,y)\rightarrow\mathtt{E}(x,y)]\wedge\forall y.[p_3(y)\rightarrow\mathtt{V}(y)]$.

Every recursion call of the transformation contributes in a result with a conjunct describing a definition for an introduced predicate. Performing the usual skolemization and writing the result in a clause form, we obtain the clauses given to the right of the definition for $[\mathtt{F}]_g^{st}$. It is easy to see that the clauses for $P_{\mathtt{F}}\wedge[\mathtt{F}]_g^{st}$ fall into the set of clauses described by the following clause schemes:

$$\begin{array}{ll}
1.\ \hat{\beta}; & \textit{where } \hat{\beta}:=\{\hat{b},\neg\hat{b}\}, \\
2.\ \neg!\hat{g}\langle !\overline{x}\rangle\vee\hat{\beta}\langle f(\overline{x}),\overline{x}\rangle. & \hat{b}:=\{\hat{a},\hat{p}_g\},\ \hat{g}:=\{\hat{a},\hat{p}_g\}
\end{array}\qquad\text{(G)}$$

We set the predicate groups $\hat{a}$ and $\hat{p}_g$ to consist from the initial predicates, and introduced definitional predicates respectively. From these groups, the auxiliary predicate groups $\hat{p}_g$, $\hat{g}$ and the literal group $\hat{\beta}$ are constructed as shown above.[2] The clauses given by the first scheme consist of propositional literals only. The second scheme represents the clauses with a *guard*: a special negative literal containing all variables $x_1,\ldots,x_n$ of the clause and no functional terms. Other literals of such a clause may contain a functional term only of the form $f'(x_1,\ldots,x_n)$ (a constant when $n=0$) that should be unique for the clause.

Saturation of the clause set. For proving decidability of the guarded fragment, we show that the clause set (G) is closed under $\mathcal{OR}_{Sel}^{\succ}$ based on a quite general class of orderings. We say that the ordering $\succ$ is *argument-monotone for predicates* if $p(t_1,\ldots,t_n)\succ q(s_1,\ldots,s_m)$ whenever $\max\{t_1,\ldots,t_n\}\succ\max\{s_1,\ldots,s_m\}$. For such orderings, in particular, $L\succ K$ for any $L\in\langle\hat{\beta}\langle !f(\overline{x}),\overline{x}\rangle\rangle$ and $K\in\langle\hat{\beta}\langle\overline{x}\rangle\rangle$. For example, any LPO-ordering based on precedence $>_P$ in which $f>_P a$ for any functional symbol (or constant) f and predicate symbol a, has the above property. From now on, assume that the resolution calculus $\mathcal{OR}_{Sel}^{\succ}$ is parametrized with some ordering $\succ$ that is argument-monotone for predicates. We chose a selection function *Sel* that selects a guard literal in every non-functional clause of the form 2 from (G). The complete case analysis of possible inferences between the clauses from (G) is given below:

[2] The predicate group $\hat{g}$ does not differ (yet) from the group $\hat{b}$. The rôle of this group will be revealed when we combine fragments and their resolution decision procedures.

1 $\hat{\beta}^*$	2 $\neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle$
1.1 $\hat{\beta} \vee \underline{\hat{b}}^*$ $:\mathtt{OR.1}$	2.1 $\neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle \vee \hat{\beta}\langle !f(\overline{x}), \overline{x}\rangle^*$
1.2 $\hat{\beta} \vee \neg\underline{\hat{b}}^*$ $:\mathtt{OR.2}$	2.1.1 $\neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle \vee \underline{\hat{b}\langle !f(\overline{x}), \overline{x}\rangle}^*$ $:\mathtt{OR.1}$
1.3 $\hat{\beta} \vee \underline{\hat{b}}^* \vee \underline{\hat{b}}:\mathtt{OF}$	2.1.2 $\neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle \vee \neg\underline{\hat{b}\langle !f(\overline{x}), \overline{x}\rangle}^*$ $:\mathtt{OR.2}$
$\mathtt{OR}[1.1; 1.2]$: $\hat{\beta}$ $:1$	2.1.3 $\neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle \vee \underline{\hat{b}\langle !f(\overline{x}), \overline{x}\rangle}^* \vee \underline{\hat{b}\langle f(\overline{x}), \overline{x}\rangle}:\mathtt{OF}$
$\mathtt{OF}[1.3]$ $: \hat{\beta} \vee \hat{b}: 1$	$\mathtt{OR}[2.1.1; 2.1.2]$: $\neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle$ $:2$
	$\mathtt{OF}[2.1.3]$ $: \neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle \vee \hat{b}\langle !f(\overline{x}), \overline{x}\rangle: 2$
	2.2 $\neg\underline{\hat{g}\langle !\overline{x}\rangle}^* \vee \neg\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle \overline{x}\rangle :\mathtt{OR.2}$
	$\mathtt{OR}[1.1; 2.2]$ $: \hat{\beta}$ $:1$
	$\mathtt{OR}[2.1.1; 2.2]$: $\neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle: 2$

The table is organized as follows. The clause schemes from (G) are spread in the table on different levels of precision. On the first level the schemes are given themselves. On the second level, different possibilities for eligible literals (marked by the asterisk) are considered. On the last level, possible inference rules that can be applied for a clause are identified and the expressions to be unified are underlined. For example, OR.1 marked to the right of the clause scheme 1.1 means that a clause represented by this scheme may act as the first premise of the ordered resolution rule. We may omit intermediate levels if there is only one case to consider, like for the clause schemes 1 and 2.2. Below the last level, inferences between preceding clauses are drawn and their conclusions are identified as instances of clause schemes. In all our decision procedures only covering expressions are unified. Therefore, the conclusion of an inference could be easily determined using Lemma 2. For example, in the inference $\mathtt{OR}[2.1.1; 2.2]$ the expressions of the form $\hat{b}\langle !f(\overline{x}), \overline{x}\rangle$ and $\hat{g}\langle \overline{y}\rangle$ are unified (we have renamed the variables apart). By Lemma 2 the unifier σ maps all variables from $\overline{x}$ to some variables $\overline{u}$. Therefore, the variable vector $\overline{y}$ is mapped to $\langle !f(\overline{u}), \overline{u}\rangle$. After the inference is made, all $\overline{u}$-variables are renamed back to $\overline{x}$-variables.

Complexity. The table above provides a very concise specification of the decision procedure for $\mathcal{GF}$. Yet, the procedure has an optimal complexity, namely 2EXPTIME. More precisely, given a guarded formula F of width w and size n, we show that the size of a saturation for F is bounded by $c_g = 2^{n \cdot 2^{O(w \log w)}}$. This bound is obtained by estimating the number of clauses from (G) that are *relevant* for F. The calculations can be found in Appendix A. A saturation of the size c can be computed in time $O(c^2)$: all possible unary and binary inferences have to be enumerated.Therefore, the procedure runs in time $2^{n \cdot 2^{O(w \log w)}}$. Similar estimation has been obtained by Grädel (1999) using model-theoretic arguments. This bound is slightly better than the one obtained by Ganzinger & de Nivelle (1999) (roughly $2^{2^{O(w^2 \log n)}}$) and by de Nivelle & de Rijke (2003) since we are using a much smaller clause class. Moreover, note, that for the *bounded-variable* guarded fragment $\mathcal{GF}^k \leftrightharpoons \{F \in \mathcal{GF} \mid wd(F) \leq k\}$ (which is relevant for knowledge representation formalisms) the procedure runs in time $2^{O(n)}$.

Theorem 3. *The resolution decision procedure for $\mathcal{GF}$ can be implemented in 2EXPTIME. For $\mathcal{GF}^k$ the procedure runs in EXPTIME.*

3.2 Deciding the Two-Variable Fragment

The *two-variable fragment* $\mathcal{FO}^2$ is the set of first-order formulas over a relational signature that are constructed using at most two variable names (say x and y):

$$\mathcal{FO}^2 ::= \mathtt{A}(x,y) \mid [\mathtt{T}_1 \wedge\!\!\!\!\vee \mathtt{T}_2](x,y) \mid \neg\mathtt{T}_1 \mid \exists x.\mathtt{T}_1(x,y) \mid \forall x.\mathtt{T}_1(x,y).$$

where $\mathtt{A}$ is an atom and $\mathtt{T}_i, i = 1,2$ are two-variable formulas. de Nivelle (2000) gave a decision procedure for $\mathcal{FO}^2$ using *lock resolution* and non-liftable orders which was extended in (de Nivelle & Pratt-Hartmann 2001) for the case with equality. The last paper employs liftable orders, but uses the lock resolution as well. In this section we present a resolution decision procedure for $\mathcal{FO}^2$ *without* using lock indexes. We enhance $\mathcal{OR}^{\succ}_{Sel}$ with an additional simplification rule, with help of which the effect of the lock resolution can be simulated.

Clause normal form translation. The definition for two-variable formulas in the negation normal form is easily obtained from the definition for $\mathcal{FO}^2$:

$$[\mathcal{FO}^2]^{nnf} ::= (\neg)\mathtt{A}(x,y) \mid [\mathtt{T}_1 \wedge\!\!\!\!\vee \mathtt{T}_2](x,y) \mid \exists x.\mathtt{T}_1(x,y) \mid \forall x.\mathtt{T}_1(x,y).$$

The structural transformation $T \Longrightarrow P_\mathtt{T} \wedge [\mathtt{T}]^{st}_t$ for $T \in [\mathcal{FO}^2]^{nnf}$ is given by:

$$\begin{array}{ll|ll}
[\mathtt{T}]^{st}_t := [(\neg)\mathtt{A}]^{st}_t : \forall\overline{x}.(P_\mathtt{T} \to (\neg)\mathtt{A}) & & 1.\ \hat{\beta}\langle\overline{x}\rangle; & |\overline{x}| \le 2; \\
\quad [\mathtt{T}_1 \wedge\!\!\!\!\vee \mathtt{T}_2]^{st}_t : \forall\overline{x}.(P_\mathtt{T} \to [P_{\mathtt{T}_1} \wedge\!\!\!\!\vee P_{\mathtt{T}_2}]) \wedge [\mathtt{T}_1]^{st}_t \wedge [\mathtt{T}_2]^{st}_t & & 2.\ \hat{\beta}\langle f(\overline{x}), \overline{x}\rangle. & |\overline{x}| \le 1; \quad (\mathrm{T}) \\
\quad [Qy.\mathtt{T}_1]^{st}_t : \forall\overline{x}.(P_\mathtt{T} \to Qy.P_{\mathtt{T}_1}) \wedge [\mathtt{T}_1]^{st}_t. & & \hat{\beta} := \{\hat{b}, \neg\hat{b}\}, \hat{b} := \{\hat{a}, \hat{p}_t\}. &
\end{array}$$

where $\overline{x} = \mathit{free}[\mathtt{T}]$, so $|\overline{x}| \le 2$ and in the last case $|\overline{x}| \le 1$. Applying skolemization to the result of the transformation, we obtain a set of clauses of the form (T), that is, non-functional clauses with at most two different variables, and clauses with at most one variable whose functional subterms are identical and contain all variables of a clause. Here $\hat{a}$ and $\hat{p}_t$ are again the predicate groups for the initial and introduced predicates respectively.

Saturation of the clause set. The important difference between the clauses from (T) and from (G) is that former are no longer guaranteed to have a guard. Therefore, the selection function *Sel* is of no particular use, for example, for the clauses ***(a)*** $\neg a(x) \vee b(x,y)$ and ***(b)*** $\neg b(x,x) \vee b(x,y)$ of the form1 as the last literal should be eligible. For the clause ***(a)***, we can make the last literal *maximal* by using orderings $\succ$ that respects arities of predicates. We say that an ordering $\succ$, that is argument-monotone for predicates, is *compatible with arities of predicates* if $p(t_1, ..., t_n) \succ q(s_1, ..., s_m)$ whenever $\max\{t_1, ..., t_n\} = \max\{s_1, ..., s_m\}$, but $n > m$. The class of *LPO*-orderings given as an example for argument-monotone orderings, can be easily restricted to fulfill this property by requiring the predicates with greater arity to be greater in precedence: $p >_P q$ if $ar(p) > ar(q)$. From now on, we assume that the $\mathcal{OR}^{\succ}_{Sel}$ is parametrized with an ordering $\succ$ that is compatible with arities of predicates.

Unfortunately the first literal in the clause ***(b)*** is maximal for *any* admissible ordering $\succ$. To deal with this problem, we introduce a new simplification rule:

Literal Projection

$$LP: \frac{[\![\ C \vee l\langle x\rangle^*\ ]\!]}{\begin{array}{c} p_{l\langle\cdot\rangle}(x) \vee C \\ \neg p_{l\langle\cdot\rangle}(x) \vee l\langle x\rangle \end{array}} \;\Bigg|\; \begin{array}{l}\textit{where (i) } l\langle x\rangle \textit{ is a non-unary} \\ \textit{literal and (ii) } C \textit{ has a non-unary} \\ \textit{predicate containing } x. \end{array}$$

This rule is an instance of a *general splitting* rule, which allows to split a clause on two by introducing a *new* predicate symbol over shared variables of its parts. Applying the rule for the clause ***(b)***, we obtain two clauses: $p_{b(\cdot,\cdot)}(x) \vee b(x,y)^*$ and $\neg p_{b(\cdot,\cdot)}(x) \vee \neg b(x,x)^*$ in which maximal literals now contain all variables. The literal projection rule (as well as the general splitting rule) is sound. It is a simplification rule for our class of orderings, as is guaranteed by the conditions of the rule. The literal projection rule extends the signature "on-the-fly", that is during a saturation process. Note, however, that this extension is always finite since the rule cannot be applied to introduced unary predicates. The possible inferences between the clauses (T) for $\mathcal{FO}^2$ are shown below:

1 $\quad \hat{\beta}\langle\overline{x}\rangle \qquad \lvert\overline{x}\rvert \leq 2$	1.3 $[\![\ !\hat{\beta}\langle x\rangle \vee !\hat{\beta}\langle y\rangle\ ]\!]$: `SP`
1.1 $\quad \hat{\beta}\langle\overline{x}\rangle \vee \hat{\beta}\langle\overline{x}\rangle^*$	`SP`[1.3]: $!\hat{\beta}\langle x\rangle$: 1 $\parallel$ $!\hat{\beta}\langle y\rangle$: 1
1.1.1 $\hat{\beta}\langle\overline{x}\rangle \vee \underline{\hat{b}\langle\overline{x}\rangle}^*$: `OR.1`	2 $\quad \hat{\beta}\langle f(\overline{x}),\overline{x}\rangle \qquad \lvert\overline{x}\rvert \leq 1$
1.1.2 $\hat{\beta}\langle\overline{x}\rangle \vee \neg\underline{\hat{b}\langle\overline{x}\rangle}^*$: `OR.2`	2.1 $\quad \hat{\beta}\langle f(\overline{x}),\overline{x}\rangle \vee \hat{\beta}\langle !f(\overline{x}),\overline{x}\rangle^*$
1.1.3 $\hat{\beta}\langle\overline{x}\rangle \vee \underline{\hat{b}\langle\overline{x}\rangle}^* \vee \underline{\hat{b}\langle\overline{x}\rangle}$: `OF`	2.1.1 $\hat{\beta}\langle f(\overline{x}),\overline{x}\rangle \vee \underline{\hat{b}\langle !f(\overline{x}),\overline{x}\rangle}^*$: `OR.1`
`OR`[1.1.1; 1.1.2]: $\hat{\beta}\langle\overline{x}\rangle$: 1	2.1.2 $\hat{\beta}\langle f(\overline{x}),\overline{x}\rangle \vee \neg\underline{\hat{b}\langle !f(\overline{x}),\overline{x}\rangle}^*$: `OR.2`
`OF`[1.1.3] : $\hat{\beta}\langle\overline{x}\rangle \vee \hat{b}\langle\overline{x}\rangle$: 1	`OR`[1.1.1; 2.1.2]: $\hat{\beta}\langle f(\overline{x}),\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}),\overline{x}\rangle$: 2
1.2 $[\![\ \hat{\beta}\langle x,y\rangle \vee !\hat{\beta}\langle !x,!y\rangle \vee \hat{\beta}\langle x\rangle^*\ ]\!]$: `LP`	`OR`[2.1.1; 1.1.2]: $\hat{\beta}\langle f(\overline{x}),\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}),\overline{x}\rangle$: 2
`LP`[1.2]: $p_{\hat{\beta}\langle\cdot\rangle}(x) \vee \hat{\beta}\langle x,y\rangle \vee !\hat{\beta}\langle !x,!y\rangle$: 1	`OR`[2.1.1; 2.1.2]: $\hat{\beta}\langle f(\overline{x}),\overline{x}\rangle$: 2
: $\neg p_{\hat{\beta}\langle\cdot\rangle}(x) \vee \hat{\beta}\langle x\rangle$: 1	2.2 $\hat{\beta}\langle\overline{x}\rangle \quad \lvert\overline{x}\rvert \leq 1$: 1

Complexity. We show that our decision procedure for the two-variable fragment has a theoretically optimal complexity, namely NEXPTIME. The saturation procedure is non-deterministic because of the splitting rule. In each branch, a saturated set of the clauses is computed that is a subset of (T). For a formula F of the size n and width w, the number of relevant clauses from (T) is bounded by $c_t = 2^{O(n \cdot 2^w)}$: see the calculations in Appendix A. Therefore, for a two-variable formula (of the width $w = 2$) the saturation is (non-deterministically) computed in the time $2^{O(n)}$. This is the same bound as has been given by Grädel, Kolaitis & Vardi (1997) who establish the small model property for $\mathcal{FO}^2$.

Theorem 4. *There is a nondeterministic resolution-based procedure that decides satisfiability of $\mathcal{FO}^2$-formulas in exponential time.*

4 Combinations of Decidable Fragments

Combinations of decidable fragments of $\mathcal{FO}$ have not been thoroughly studied in literature. Some (un)decidability results about boolean combination of *prefix-vocabulary* classes are mentioned in (Börger et al. 1997). We introduce the notion of *recursive composition* of decidable fragments. This notion has been inspired by the fact that decidable logical formalisms, such as description logics, CTL, PDL, etc., are usually defined recursively. They can be viewed as collection of "safe constructors" for formulas of the interest. The natural idea is, therefore, to combine these constructors to obtain more expressive formalisms.

4.1 Combining the Guarded and Two-Variable Fragments

We define a *recursive composition* $\mathcal{GF}|\mathcal{FO}^2$ of the guarded and two-variable fragments by *joining* their recursive definitions in the following way:

$$\begin{array}{ll}
\mathcal{GF}|\mathcal{FO}^2 \ ::= \mathtt{A} \mid \mathtt{F}_1 \mid \mathtt{T}_1; & \textit{where } \mathtt{A},\, \mathtt{G} \textit{ are atoms},\\
\mathcal{GF} ::= \mathtt{H} \mid \mathtt{F}_1 {\wedge\!\!\!\vee} \mathtt{F}_2 \mid \neg\mathtt{F}_1 \mid \forall\overline{x}.(\mathtt{G}\rightarrow\mathtt{F}_1) \mid \exists\overline{x}.(\mathtt{G}\wedge\mathtt{F}_1); & \mathtt{H}\in\mathcal{GF}|\mathcal{FO}^2,\ \mathtt{F}_i\in\mathcal{GF}\\
\mathcal{FO}^2 ::= \mathtt{H}(x,y) \mid [\mathtt{T}_1 {\wedge\!\!\!\vee} \mathtt{T}_2](x,y) \mid \neg\mathtt{T}_1 \mid Qx.\mathtt{T}_1(x,y). & \textit{and } \mathtt{T}_i\in\mathcal{FO}^2,\ i=1,2.
\end{array}$$

One could see that this definition can be simplified in the following way:

$$\mathcal{GF}|\mathcal{FO}^2 \ ::= \mathtt{A} \mid \mathtt{H}_1 {\wedge\!\!\!\vee} \mathtt{H}_2 \mid \neg\mathtt{H}_1 \mid \forall\overline{x}.(\mathtt{G}\rightarrow\mathtt{H}_1) \mid \exists\overline{x}.(\mathtt{G}\wedge\mathtt{H}_1) \mid Qx.\mathtt{H}_1(x,y).$$

where $\mathtt{H}_i \in \mathcal{GF}|\mathcal{FO}^2, i = 1,2$; however, we shell address to the first definition to demonstrate the general approach. The definition extends $\mathcal{GF}$ and $\mathcal{FO}^2$ by essentially allowing to use guarded formulas with at most two free variables as two-variable atoms as well as two-variable formulas as atoms in guarded formulas. For example, the following formula belongs to $\mathcal{GF}|\mathcal{FO}^2$ but to none of its subfragment: $TG(y) := \forall x.(a(x) \vee \{\forall z.(c(x,y,z)\rightarrow[\forall u.b(y,u)]_{:\mathcal{FO}^2})\}^{:\mathcal{GF}})_{:\mathcal{FO}^2}$.

Deciding the combined fragment. For deciding a recursive combination of fragments, it is possible to reuse resolution decision procedures for its components. The negation normal form and structural transformations fir $\mathcal{GF}|\mathcal{FO}^2$ are obtained by *joining* the respective transformations for $\mathcal{GF}$ and $\mathcal{FO}^2$:[3]

$$\begin{array}{ll}
[\mathcal{GF}|\mathcal{FO}^2]^{nnf} := (\neg)\mathtt{A} \mid \mathtt{F}_1 \mid \mathtt{T}_1. & [\mathtt{F}]^{st}_g := [\mathtt{F}_1 {\wedge\!\!\!\vee} \mathtt{F}_2]^{st}_g : \forall\overline{x}.(P_{\mathtt{F}}\rightarrow[P_{\mathtt{F}_1} {\wedge\!\!\!\vee} P_{\mathtt{F}_2}])\wedge[\mathtt{F}_1]^{st}_g\wedge[\mathtt{F}_2]^{st}_g \mid\\
[\mathcal{GF}]^{nnf}_w ::= \mathtt{H} \mid \mathtt{F}_1 {\wedge\!\!\!\vee} \mathtt{F}_2 \mid & \quad [\forall\overline{y}.(\mathtt{G}\rightarrow\mathtt{F}_1)]^{st}_g : \forall\overline{x}.(P_{\mathtt{F}}\rightarrow\forall\overline{y}.[\mathtt{G}\rightarrow P_{\mathtt{F}_1}])\wedge[\mathtt{F}_1]^{st}_g \mid\\
\qquad \forall\overline{x}.(\mathtt{G}\rightarrow\mathtt{F}_1) \mid \exists\overline{x}.\mathtt{F}_1. & \quad [\exists y.\mathtt{F}_1]^{st}_g : \forall\overline{x}.(P_{\mathtt{F}}\rightarrow\exists y.P_{\mathtt{F}_1})\wedge[\mathtt{F}_1]^{st}_g \mid\\
[\mathcal{FO}^2]^{nnf} ::= \mathtt{H}(x,y) \mid & \quad [\mathtt{H}]^{st}_g : \forall\overline{x}.(P_{\mathtt{F}}\rightarrow P_{\mathtt{H}})\wedge[\mathtt{H}]^{st}_{gt}.\\
\qquad [\mathtt{T}_1 {\wedge\!\!\!\vee} \mathtt{T}_2](x,y) \mid Qx.\mathtt{T}(x,y). & [\mathtt{T}]^{st}_t := [\mathtt{T}_1 {\wedge\!\!\!\vee} \mathtt{T}_2]^{st}_t : \forall\overline{x}.(P_{\mathtt{T}}\rightarrow[P_{\mathtt{T}_1} {\wedge\!\!\!\vee} P_{\mathtt{T}_2}])\wedge[\mathtt{T}_1]^{st}_t\wedge[\mathtt{T}_2]^{st}_t \mid\\
[\mathtt{H}]^{st}_{gt} := [\mathtt{F}_1]^{st}_{gt} : [\mathtt{F}_1]^{st}_g \mid [\mathtt{T}_1]^{st}_{gt} : [\mathtt{T}_1]^{st}_t \mid & \quad [Qy.\mathtt{T}_1]^{st}_t : \forall\overline{x}.(P_{\mathtt{T}}\rightarrow Qy.P_{\mathtt{T}_1})\wedge[\mathtt{T}_1]^{st}_t \mid\\
\qquad [(\neg)\mathtt{A}]^{st}_{gt} : \forall\overline{x}.(P_{\mathtt{H}}\rightarrow(\neg)\mathtt{A}). & \quad [\mathtt{H}]^{st}_t : \forall\overline{x}.(P_{\mathtt{T}}\rightarrow P_{\mathtt{H}})\wedge[\mathtt{H}]^{st}_{gt}.
\end{array}$$

$$\textit{where } \mathtt{A} \textit{ is an atom},\ \mathtt{H}\in[\mathcal{GF}|\mathcal{FO}^2]^{nnf},\ \mathtt{F}_i\in[\mathcal{GF}]^{nnf}_w,\ \mathtt{T}_i\in[\mathcal{FO}^2]^{nnf},\ i=1,2.$$

Example 3. The structural transformation for $TG(y)$ yields: $p^t_0(y)\wedge\forall x.[p^t_0(y)\rightarrow\forall x.(a(x)\vee p^t_1(x,y))]\wedge\boldsymbol{\forall xy.[p^t_1(x,y)\rightarrow p^g_1(x,y)]}\wedge\forall xy.[p^g_1(x,y)\rightarrow\forall z.(c(x,y,z)\rightarrow p^g_2(y))]\wedge\boldsymbol{\forall y.[p^g_2(y)\rightarrow p^t_2(y)]}\wedge\forall y.[p^t_2(y)\rightarrow\forall u.p^t_3(y,u)]\wedge\forall yu.[p^t_3(y,u)\rightarrow b(y,u)]$.

The transformations are connected in *base* cases, where interaction between fragments occurs. The clause class for the combined fragment $\mathcal{GF}|\mathcal{FO}^2$ is the *union* of clause classes (G) and (T) with additional "linking" clauses of the form $\neg!\hat{p}_g(\overline{x})\vee\hat{p}_t(\overline{x})$ and $\neg!\hat{p}_t(\overline{x})\vee\hat{p}_g(\overline{x})$. Linking clauses can be captured if we unite $\hat{b}$ and $\hat{\beta}$-groups for both clause classes (G) and (T) (but keeping the $\hat{g}$-group as it was!). This extension does not produce new inferences inside each individual clause class. On the other hand, inferences *between* (G) and (T) can be shown to fall into either (G) or (T). This proofs decidability of $\mathcal{GF}|\mathcal{FO}^2$:

[3] The definition for $\mathcal{GF}|\mathcal{FO}^2$ is given "bottom-up", i.e. started from the base cases; for transformations we need to apply definitions "top-down", i.e. ending with base cases, to ensure that recursion terminates.

$\texttt{OR}[\texttt{T}.1.1.1;\texttt{G}.2.1.2]\colon \neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}),\overline{x}\rangle\colon\texttt{G}.2$	$\texttt{OR}[\texttt{T}.2.1.1;\texttt{G}.2.1.2]\colon \neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}),\overline{x}\rangle\colon\texttt{G}.2$
$\texttt{OR}[\texttt{G}.2.1.1;\texttt{T}.1.1.2]\colon \neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}),\overline{x}\rangle\colon\texttt{G}.2$	$\texttt{OR}[\texttt{G}.2.1.1;\texttt{T}.2.1.2]\colon \neg!\hat{g}\langle !\overline{x}\rangle \vee \hat{\beta}\langle f(\overline{x}),\overline{x}\rangle\colon\texttt{G}.2$
$\texttt{OR}[\texttt{T}.1.1.1;\texttt{G}.2.2]\ \colon\hat{\beta}\langle\overline{x}\rangle \qquad \lvert\hat{x}\rvert \le 2\colon\texttt{T}.1$	$\texttt{OR}[\texttt{T}.2.1.1;\texttt{G}.2.2]\ \colon\hat{\beta}\langle f(\overline{x}),\overline{x}\rangle \qquad \lvert\overline{x}\rvert \le 1\colon\texttt{T}.2$

Complexity. The fragment $\mathcal{GF}|\mathcal{FO}^2$ is decided by the non-deterministic procedure computing a saturation that is a subset of (G)+(T). Thus, the size of any saturation and (non-deterministic) time to compute it, are bounded by $c_{gt}\!:=c_g+c_t = 2^{n\cdot 2^{O(w\log w)}}$ for n being the size of a formula and w its width (see the calculations before). From this, we immediately obtain that our procedure is in 2**N**EXPTIME. However, more careful analysis shows that it could be implemented in 2EXPTIME. Indeed, on every saturation branch at most $c_{\texttt{SP}} = 2^{O(n)}$ choices are maid, since we split only clauses of the from $\texttt{T}.1.3$. Therefore, the size of the computation tree is bounded by $2^{c_{\texttt{SP}}}\cdot c_{gt} = 2^{2^{O(n)}}$. Thus, saturation with backtracking decides $\mathcal{GF}|\mathcal{FO}^2$ in 2EXPTIME. Note also, that when the width w is bounded, the procedure runs in NTIME($2^{O(n)}$).

Theorem 5. *The resolution-based decision procedure for $\mathcal{GF}|\mathcal{FO}^2$ can be implemented in 2EXPTIME. $\mathcal{GF}^k|\mathcal{FO}^2$ can be decided in NEXPTIME.*

4.2 Combinations With the Monadic Fragment

We demonstrate how our method can be extended to obtain decidability results for combinations with the full monadic fragment. The *monadic fragment* of $\mathcal{FO}$ (also called *Löwenheim class*) is defined as the set of $\mathcal{FO}$-formulas constructed from one-variable atoms:

$\mathcal{M} ::= \texttt{A}(x) \mid \texttt{M}_1 \wedge\!\!\!\!\vee \texttt{M}_2 \mid \neg\texttt{M}_1 \mid Qx.\texttt{M}_1.$ *where* $\texttt{A}$ *is an atom,* $\texttt{M}_i \in \mathcal{M}, i = 1,2.$

The *full monadic fragment* (also known as *Löb-Gurevich class*) is an extension of $\mathcal{M}$ where in addition *unary functional symbols* are allowed in formulas:

$\mathcal{M}_f ::= \texttt{A}(x) \mid \texttt{M}_1(x)[f(x)/x] \mid \texttt{M}_1 \wedge\!\!\!\!\vee \texttt{M}_2 \mid \neg\texttt{M}_1 \mid Qx.\texttt{M}_1.$

For example, the following formula $\exists y.(b(y) \wedge \forall z.[a(z) \vee b(f(y))]) \in \mathcal{M}_f$

Joyner Jr. (1976) presented the first resolution-based decision for the monadic class. We give a resolution decision procedure for the *full* monadic class and we show that our procedure has the theoretically optimal complexity.

Clause normal form translation. The negation normal form for monadic formulas and their structural transformation are defined below:

$[\mathcal{M}_f]^{nnf} ::= (\neg)\texttt{A}(x) \mid \texttt{M}_1(x)[f(x)/x] \mid \texttt{M}_1 \wedge\!\!\!\!\vee \texttt{M}_2 \mid Qx.\texttt{M}.$

$$\begin{array}{rll}
[\texttt{M}]^{st}_m\colon= & & 1.\ \hat{\beta}\langle f(\overline{x})\rangle \vee \hat{\beta}\langle\overline{x}\rangle;\quad \lvert\overline{x}\rvert \le 1 \qquad \text{(M)}\\
[(\neg)\texttt{A}(x)]^{st}_m\colon \forall x.(P_{\texttt{M}} \rightarrow (\neg)\texttt{A}(x)) & \mid & 2.\ \hat{\rho}_m(\overline{x},y) \vee \hat{\rho}_m(\overline{x}) \vee \hat{\rho}^1_m\langle\overline{x},y\rangle;\\
[\texttt{M}_1[f(x)/x]]^{st}_m\colon \forall x.(P_{\texttt{M}} \rightarrow P_{\texttt{M}_1}[f(x)/x]) \wedge [\texttt{M}_1]^{st}_m & \mid & 3.\ \hat{\rho}_m(\overline{x},f(\overline{x})) \vee \hat{\rho}_m(\overline{x}) \vee \hat{\rho}^1_m\langle\overline{x},f(\overline{x})\rangle;\\
[\texttt{M}_1 \wedge\!\!\!\!\vee \texttt{M}_2]^{st}_m\colon \forall\overline{x}.(P_{\texttt{M}} \rightarrow [P_{\texttt{M}_1} \wedge\!\!\!\!\vee P_{\texttt{M}_2}]) \wedge [\texttt{M}_1]^{st}_m \wedge [\texttt{M}_2]^{st}_m & \mid & 4.\ \hat{\rho}_m(\overline{x}) \vee \hat{\beta}\langle f(\overline{x})\rangle \vee \hat{\rho}^1_m\langle\overline{x}\rangle;\\
[Qy.\texttt{M}_1]^{st}_m\colon \forall\overline{x}.(P_{\texttt{M}} \rightarrow Qy.P_{\texttt{M}_1}) \wedge [\texttt{M}_1]^{st}_m. & & \hat{\beta} := \{\hat{b},\neg\hat{b}\}, \hat{\rho}_m := \{\hat{p}_m,\neg\hat{p}_m\}, \hat{b} := \{\hat{a},\hat{p}_m\}
\end{array}$$

Besides introducing definitional predicates, the structural transformation also *flattens* atoms with nested functional terms using additional predicate names. A definitional predicate $P_{\mathtt{M}'}$ is either unary, when $\mathtt{M}'$ corresponds two the first two cases of the definitions, or, for the other cases, variables of $P_{\mathtt{M}'}$ are all variables in the scope of which $\mathtt{M}'$ occurs in a monadic formula and arranged according to their first appearance.[4] The order of variables in definitional predicates is crucial for showing decidability of $\mathcal{M}_f$ by resolution. Therefore, it is taken into account in the definition of the clause class (M), which captures the result of the transformation. Here $\hat{\rho}_m$ is again the group of the definitional predicates and $\hat{\rho}^1_m$ is the subgroup of $\hat{\rho}_m$ consisting of *unary* literal symbols. The complete case analysis of possible inferences between the clauses from (M) is given below. Note, that the ordered factoring rule is never applied because of the fixed order of variables in arguments (if two literals can be factored then they are equal):

1 $\hat{\beta}\langle f(\overline{x})\rangle \vee \hat{\beta}\langle \overline{x}\rangle \qquad \lvert\overline{x}\rvert \le 1$	3 $\hat{\rho}_m(\overline{x}, f(\overline{x})) \vee \hat{\rho}_m(\overline{x}) \vee \hat{\rho}^1_m\langle \overline{x}, f(\overline{x})\rangle$
1.1 $\hat{\beta}\langle f(\overline{x})\rangle \vee \hat{\beta}\langle \overline{x}\rangle \vee \underline{\hat{\beta}\langle ! f(\overline{x})\rangle}^{*}$: `OR`	3.1 $\underline{\hat{\rho}_m(\overline{x}, f(\overline{x}))}^{*} \vee \hat{\rho}_m(\overline{x}) \vee \hat{\rho}^1_m\langle \overline{x}, f(\overline{x})\rangle$: `OR`
1.2 $\hat{\beta}\langle \overline{x}\rangle \vee \underline{\hat{\beta}\langle ! \overline{x}\rangle}^{*}$: `OR`	3.2 $\hat{\rho}_m(\overline{x}) \vee \hat{\rho}^1_m\langle \overline{x}, f(\overline{x})\rangle$: 4
`OR`$[1.*; 1.*]$: 1	`OR`$[1.*, 3.1]$: 1 `OR`$[2.*, 3.1]$: 3 `OR`$[3.1, 3.1]$: 3
2 $\hat{\rho}_m(\overline{x}, y) \vee \hat{\rho}_m(\overline{x}) \vee \hat{\rho}^1_m\langle \overline{x}, y\rangle$	4 $\hat{\rho}_m(\overline{x}) \vee \hat{\beta}\langle f(\overline{x})\rangle \vee \hat{\rho}^1_m\langle \overline{x}\rangle$
2.1 $\underline{\hat{\rho}_m(\overline{x}, y)}^{*} \vee \hat{\rho}_m(\overline{x}) \vee \hat{\rho}^1_m\langle \overline{x}, y\rangle$: `OR`	4.1 $\hat{\rho}_m(\overline{x}) \vee \hat{\beta}\langle f(\overline{x})\rangle \vee \hat{\rho}^1_m\langle \overline{x}\rangle \vee \underline{\hat{\beta}\langle ! f(\overline{x})\rangle}^{*}$: `OR`
2.2 $\underline{\hat{\rho}_m(\overline{x})}^{*} \vee \hat{\rho}^1_m\langle \overline{x}\rangle$: `OR`	4.2 $\hat{\rho}_m(\overline{x}) \vee \hat{\rho}^1_m\langle \overline{x}\rangle$: 2
2.3 $[\![\ \hat{\rho}_m(\overline{x}) \vee \hat{\rho}^1_m\langle \overline{x}\rangle \vee !\hat{\rho}^1_m\langle y\rangle\ ]\!]$: `SP`	`OR`$[1.1, 4.1]$: 1 `OR`$[1.2, 4.1]$: 4 `OR`$[2.1, 4.1]$: 4
`OR`$[1.*, 2.*]$: 1 `OR`$[2.*, 2.*]$: 2	`OR`$[2.2, 4.1]$: 4 `OR`$[3.1, 4.1]$: 1 `OR`$[4.1, 4.1]$: 4
`SP`$[2.3]$: $\hat{\rho}_m\langle \overline{x}\rangle \vee \hat{\rho}^1_m\langle \overline{x}\rangle$: 2 $\parallel$ $!\hat{\rho}^1_m\langle y\rangle$: 1	

Complexity. By estimating the size of (M) one can immediately obtain that the complexity of our decision procedure is NTIME $(2^{O(n^2)})$. With more careful analysis it is possible to obtain the best known bound NTIME $(2^{O(n)})$ (see Börger et al. 1997). The calculations are given in Appendix A.

Theorem 6. *There is a nondeterministic saturation-based procedure that decides satisfiability of $\mathcal{M}_f$ in exponential time.*

Combining the monadic, two-variable and guarded fragments. The general scheme for combining fragments looks as follows. First, the base cases in recursive definitions of fragments are modified. Then a new fragment $\mathcal{F}$ is defined by linking them as follows:

$$\mathcal{GF}{::=}\mathtt{H} \mid \mathtt{F}_1 \bowtie \mathtt{F}_2 \mid \neg\mathtt{F}_1 \mid \forall\overline{x}.(\mathtt{G}\rightarrow\mathtt{F}_1) \mid \exists\overline{x}.(\mathtt{G}\wedge\mathtt{F}_1). \quad \boldsymbol{\mathcal{F}}: \quad \mathcal{GF}|\mathcal{M}_f{::=}\mathtt{A} \mid \mathtt{F} \mid \mathtt{M}.$$

$$\mathcal{FO}^2{::=}\mathtt{H}(x,y) \mid [\mathtt{T}_1 \bowtie \mathtt{T}_2](x,y) \mid \neg\mathtt{T}_1 \mid Qx.\mathtt{T}(x,y). \qquad \mathcal{FO}^2|\mathcal{M}_f{::=}\mathtt{A} \mid \mathtt{T} \mid \mathtt{M}.$$

$$\mathcal{M}_f{::=}\mathtt{H}(x) \mid \mathtt{M}_1(x)[f(x)/x] \mid \mathtt{M}_1 \bowtie \mathtt{M}_2 \mid \neg\mathtt{M}_1 \mid Qx.\mathtt{M}. \quad \mathcal{GF}|\mathcal{FO}^2|\mathcal{M}_f{::=}\mathtt{A} \mid \mathtt{F} \mid \mathtt{T} \mid \mathtt{M}.$$

$$\textit{where } \mathtt{H}\in\mathcal{F},\ \mathtt{F}_i\in\mathcal{GF},\ \mathtt{T}_i\in\mathcal{FO}^2 \textit{ and } \mathtt{M}_i\in\mathcal{M}_f.$$

For deciding $\mathcal{F}$ using our approach, the sequence of transformations is applied that can be obtained by "joining" the transformations for the components of $\mathcal{F}$. In particular it can be shown similarly as for $\mathcal{GF}|\mathcal{FO}^2$ that the clause class for a combined fragment is the union of the clause classes for its parts. Decidability

[4] Without loss of generality we assume that the input monadic formula is a sentence.

of $\mathcal{GF}|\mathcal{M}_f$, $\mathcal{FO}|\mathcal{M}_f$ and $\mathcal{GF}|\mathcal{FO}^2|\mathcal{M}_f$ is proven by examining all possible inferences between clause classes (G), (T) and (M):

$\mathtt{OR}[\mathtt{M.1.1}, \mathtt{G.2.1}]\colon \mathtt{G.2}$	$\mathtt{OR}[\mathtt{M.2.2}; \mathtt{G.2.1}]\colon \mathtt{G.2}$	$\mathtt{OR}[\mathtt{M.2.1}, \mathtt{T.1.1}]\colon \mathtt{T.1}$	$\mathtt{OR}[\mathtt{M.3.1}, \mathtt{T.1.1}]\colon \mathtt{T.2}$
$\mathtt{OR}[\mathtt{M.1.1}; \mathtt{G.2.2}]\colon \mathtt{M.1}$	$\mathtt{OR}[\mathtt{M.3.1}; \mathtt{G.2.1}]\colon \mathtt{G.2}$	$\mathtt{OR}[\mathtt{M.2.1}, \mathtt{T.2.1}]\colon \mathtt{T.2}$	$\mathtt{OR}[\mathtt{M.3.1}, \mathtt{T.2.1}]\colon \mathtt{T.2}$
$\mathtt{OR}[\mathtt{M.1.2}; \mathtt{G.2.1}]\colon \mathtt{G.2}$	$\mathtt{OR}[\mathtt{M.4.1}; \mathtt{G.2.1}]\colon \mathtt{G.2}$	$\mathtt{OR}[\mathtt{M.2.2}, \mathtt{T.1.1}]\colon \mathtt{T.1}$	$\mathtt{OR}[\mathtt{M.4.1}, \mathtt{T.1.1}]\colon \mathtt{M.4}$
$\mathtt{OR}[\mathtt{M.1.2}; \mathtt{G.2.2}]\colon \mathtt{M.1}$	$\mathtt{OR}[\mathtt{M.4.1}; \mathtt{G.2.2}]\colon \mathtt{M.4}$	$\mathtt{OR}[\mathtt{M.2.2}, \mathtt{T.2.1}]\colon \mathtt{T.2}$	$\mathtt{OR}[\mathtt{M.4.1}, \mathtt{T.2.1}]\colon \mathtt{T.2}$
$\mathtt{OR}[\mathtt{M.2.1}; \mathtt{G.2.1}]\colon \mathtt{G.2}$			

Example 4. The formula $MG^2(x) := \forall y.(b(x,y) \rightarrow [a(y) \vee \{a(s(x)) \wedge \neg b(x,x)\}]) \in \mathcal{GF}^2|\mathcal{M}$. The structural transformation for $MG^2(x)$ produce: $p_0^g(x) \wedge \forall x.(p_0^g(x) \rightarrow \forall y.[b(x,y) \rightarrow p_1^g(x,y)]) \wedge \boldsymbol{\forall xy.[p_1^g(x,y) \rightarrow p_1^m(x,y)]} \wedge \forall xy.[p_1^m(x,y) \rightarrow (p_2^m(y) \vee p_3^m(x,\boldsymbol{y}))] \wedge \forall y.[p_2^m(y) \rightarrow a(y)] \wedge \forall x.[p_3^m(x,y) \rightarrow p_4(x)] \wedge \forall x.[p_3^m(x,y) \rightarrow p_5(x)] \wedge \forall x.[p_4(x) \rightarrow p_6(s(x))] \wedge \forall x.[p_6(x) \rightarrow a(x)] \wedge \forall x.[p_5(x) \rightarrow \neg b(x,x)]$.

Theorem 7. *There are resolution-based decision procedures for the following fragments of the listed complexity:*

$\mathcal{GF}|\mathcal{M}_f$: 2EXPTIME; $\mathcal{FO}^2|\mathcal{M}_f$: NEXPTIME; $\mathcal{GF}|\mathcal{FO}^2|\mathcal{M}_f$: 2EXPTIME;
$\mathcal{GF}^k|\mathcal{M}_f$: NEXPTIME; $\mathcal{GF}^k|\mathcal{FO}^2|\mathcal{M}_f$: NEXPTIME.

Proof. The details of the proof can be found in Appendix B. □

5 Undecidability Results

In this section we show that many of the decidability results given in the paper cannot be carried out to the case with equality. More precisely, we show that already the fragments $\mathcal{GF}^3_{\simeq}|\mathcal{FO}^2$, $\mathcal{GF}^3|\mathcal{FO}^2_{\simeq}$ and $\mathcal{GF}^3|\mathcal{FO}^2|\mathcal{M}_{\simeq}$ are undecidable. We prove that by a reduction from the satisfiability problem for the Goldfarb class. The *Goldfarb class* is the set of first-order formulas with equality having the quantifier-prefix $\forall^2\exists$ (see Börger et al. 1997). It forms a conservative reduction class, so, in particular it is undecidable. For any sentence $F' = \forall xy.\exists z.F$ from the Goldfarb class (where F is quantifier-free), consider the sentence:

$$F_{GT} := \forall xy.p_1(x,y) \wedge \forall xy.[p_1(x,y) \rightarrow \exists z.p_2(x,y,z)] \wedge \forall xyz.[p_2(x,y,z) \rightarrow F]$$

Let F_{GT}^- is obtained from F_{GT} by replacing every occurrence of equality with a fresh binary predicate $E(x,y)$ and let $F_E := \forall xy.[E(x,y) \leftrightarrow x \simeq y]$ be the "definition" for $E(x,y)$. Then ***(i)*** F' is (finitely) satisfiable *iff* ***(ii)*** F_{GT} is (finitely) satisfiable *iff* ***(iii)*** $F_{GT}^- \wedge F_E$ is (finitely) satisfiable. Note, that F_{GT}^- is a conjunction of the two-variable formula and the three-variable guarded formulas (without equality!). Finally, observe that F_E is expressible in every of the fragments $\mathcal{GF}^2_{\simeq}$, $\mathcal{FO}^2_{\simeq}$ and $\mathcal{FO}^2|\mathcal{M}_{\simeq}$. Therefore, the formula $F_{GT}^- \wedge F_E$ is expressible in *all* fragments $\mathcal{GF}^3_{\simeq}|\mathcal{FO}^2$, $\mathcal{GF}^3|\mathcal{FO}^2_{\simeq}$ and $\mathcal{GF}^3|\mathcal{FO}^2|\mathcal{M}_{\simeq}$. This construction $F' \Rightarrow F_{GT}^- \wedge F_E$ provides us a reduction from the Goldfarb class.

Theorem 8. *The fragments $\mathcal{GF}^3_{\simeq}|\mathcal{FO}^2$, $\mathcal{GF}^3|\mathcal{FO}^2_{\simeq}$ and $\mathcal{GF}^3|\mathcal{FO}^2|\mathcal{M}_{\simeq}$ form conservative reduction classes.*

6 Conclusions

We have shown how ordered resolution enhanced with the schematic notation can be used to specify decision procedures for many interesting fragments and their combinations, almost directly from the recursive definitions of fragments. The limitations of our combinational approach are yet to be explored, in particular, the decidability statuses for the fragments $\mathcal{GF}_{\simeq}|\mathcal{M}_{\simeq}$ and $\mathcal{FO}^2_{\simeq}|\mathcal{M}_{\simeq}$.

References

Armando, A., Ranise, S. & Rusinowitch, M. (2001), 'Uniform derivation of decision procedures by superposition', *Lecture Notes in Computer Science* **2142**, 513+.

Bachmair, L. & Ganzinger, H. (2001), Resolution theorem proving, *in* A. Robinson & A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 2, pp. 19–99.

Bachmair, L., Ganzinger, H. & Waldmann, U. (1993), Set constraints are the monadic class, *in* 'Eighth Annual IEEE Symposium on Logic in Computer Science', IEEE, Montreal, Canada, pp. 75–83.

Börger, E., Grädel, E. & Gurevich, Y. (1997), *The Classical Decision Problem*, Perspectives of Mathematical Logic, Springer-Verlag. Second printing (Universitext) 2001.

de Nivelle, H. (2000), An overview of resolution decision procedures, *in* M. Faller, S. Kaufmann & M. Pauly, eds, 'Formalizing the Dynamics of Information', Vol. 91 of *CSLI Publications*, Center for the Study of Language and Information, Stanford University, Palo Alto, USA, pp. 115–130.

de Nivelle, H. & de Rijke, M. (2003), 'Deciding the guarded fragments by resolution', *Journal of Symbolic Computation* **35**, 21–58.

de Nivelle, H. & Pratt-Hartmann, I. (2001), A resolution-based decision procedure for the two-variable fragment with equality., *in* T. N. R. Goré, A. Leitsch, ed., 'In: Proc. 1st Int. Joint Conf. on Automated Reasoning (IJCAR-2001)', Vol. 2083 of *Lect. Notes Artif. Intell.*, Springer, Berlin, pp. 211–225.

Fermüller, C., Leitsch, A., Hustadt, U. & Tammet, T. (2001), Resolution decision procedures, *in* A. Robinson & A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 25, pp. 1791–1849.

Fermüller, C., Leitsch, A., Tammet, T. & Zamov, N. (1993), *Resolution Methods for the Decision Problem*, Vol. 679 of *LNAI*, Springer, Berlin, Heidelberg.

Ganzinger, H. & de Nivelle, H. (1999), A superposition decision procedure for the guarded fragment with equality, *in* 'Proc. 14th IEEE Symposium on Logic in Computer Science', IEEE Computer Society Press, pp. 295–305.

Grädel, E. (1999), 'On the restraining power of guards', *Journal of Symbolic Logic* **64(4)**, 1719–1742.

Grädel, E. & Otto, M. (1999), 'On logics with two variables', *Theoretical Computer Science* **224**, 73–113.

Grädel, E., Kolaitis, P. & Vardi, M. (1997), 'On the Decision Problem for Two-Variable First-Order Logic', *Bulletin of Symbolic Logic* **3**, 53–69.

Joyner Jr., W. H. (1976), 'Resolution strategies as decision procedures', *Journal of the ACM* **23**(3), 398–417.

Lutz, C., Sattler, U. & Wolter, F. (2001), Modal logics and the two-variable fragment, *in* 'Annual Conference of the European Association for Computer Science Logic CSL'01', LNCS, Springer Verlag, Paris, France.

Appendix A
Complexity Calculations

In this appendix we give a detailed estimation for the complexity bounds for the guarded fragment, two-variable fragment and the monadic fragment that were sketched in the main part of the paper.

A.1 A Complexity Bound for the Guarded Fragment

Given a guarded formula F of the size n and width w, we estimate the number of clauses from (G), that can appear in saturation inferences for F. We call these clauses the *relevant clauses* for F. In particular, these clauses can be constructed only from the predicate symbols occurring in F and additional definitional predicates and Skolem functions that were introduced during the CNF transformation. Assume that after the CNF transformation for F we have c_0 clauses over n_1 predicate symbols and n_2 Skolem functions. Let a_1 and a_2 be the maximal arities of predicate and functional symbols respectively. Note that: ***(i)*** the number of initial clauses $c_0 = O(n)$ and each initial clause has at most w variables and consists of at most three literals; ***(ii)*** $max(n_1, n_2) = O(n)$, $max(a_1, a_2) \leq n$.

The number of relevant clauses from (G) can be estimated as follows. The number of clauses given by the first scheme is bounded by $c_1 := 2^{2n_1} = 2^{O(n)}$ as there are at most $2n_1$ different propositional literals. Each clause C_2 given by the second scheme can be represented by a functional term $f(\overline{x})$ and function-free clause over at most $w + 1$ variables: we replace all occurrences of $f(\overline{x})$ with some new variable[5]. The number of different terms of the form $f(\overline{x})$ having at most w different variables is bounded by $c_t := n_2 \cdot w^{a_2}$. The number of different functional-free literals over $w + 1$ variables, similarly, does not exceed $c_l := 2n_1 \cdot (w + 1)^{a_1}$. Therefore, the number of relevant clauses represented by the second scheme is bounded by $c_2 := c_t \cdot 2^{c_l} = 2^{O(n \cdot (w+1)^n)}$, since $a_1 \leq n$. Unfortunately, this is not satisfactory result, as we obtain the double exponential bound on the number of clauses even with the fixed width w.

The main source of the complexity is the maximal arity of predicate symbols, which in principle can be linear in n. It is possible to translate the formula F preserving satisfiability to a formula having predicates whose arity is at most w, like it is done for the two-variable fragment in (Grädel et al. 1997). However, the same complexity bounds can be achieved without using additional transformations. Note that every literal in the clause that can appear in the saturation process, is an instance of some literal in an initial clause obtained after the CNF transformation. There are at most $n_l := 3 \cdot c_0 = O(n)$ different literals that can occur in initial clauses and *with at most w variables.* Every literal in the clause C_2 is obtain from some of these literals by substituting variables from $\overline{x}$ or the term $f(\overline{x})$ for every variable of the literal. Therefore, the number of different

[5] Remember, that the number of variables does not grow in the inferences, so every relevant clause contains at most w variables.

literals that may occur C_2 is bounded by $c_l' := n_l \cdot (w+1)^{\boldsymbol{w}}$ and the number of relevant clauses for F is bounded by $c_g := 2^{n \cdot 2^{O(w \log w)}}$.

A.2 A Complexity Bound for the Two-Variable Fragment

We estimate the maximal number c_t of relevant clauses from (T) for a formula F of the size n and width w, where F is not necessarily a two-variable formula. We define c_t to be the number of clauses of the form (T) that can be obtained by saturating the *initial clauses* for F. As in the previous section, we assume that ***(i)*** the number of the initial clauses for F is $c_0 \leq n$ and they contain at most w variables and at most three literals each; ***(ii)*** $max(n_1, n_2) = O(n)$, $max(a_1, a_2) \leq n$ for n_1, n_2 being the number of possible predicate and functional symbols and a_1 and a_2 being their maximal arities respectively. In particular, these assumptions hold for a two-variable formula since the CNF transformation is linear in n and the literal projection rule may introduce at most $n_p = O(n)$ new unary predicates.

Let n_l be the number of literals that occur in the initial clauses. It follows from the conditions above that $n_l = O(n)$ and each initial literal contain at most w variables. The number of the clauses given by the first scheme from (T) is bounded by $c_1 := 2^{(n_l+n_p)\cdot 2^w}$ and the number of the clauses given by the second scheme is bounded by $c_2 := 2^{n_2} \cdot 2^{(n_l+n_p)\cdot 2^w}$ (taking into account the introduced predicates). So, the possible number of the relevant clauses for F is bounded by $c_t := c_1 + c_2 = 2^{O(n \cdot 2^w)}$.

A.3 A Complexity Bound for the Full Monadic Fragment

To give a bound on the size of a saturation for a monadic formula, we estimate the number of clauses from (M) that can be constructed using n_1 predicate symbols and n_2 functional symbols with the maximal arities a_1 and a_2 respectively. As usual, we may assume, that $max(n_1, n_2) = O(n)$ and $max(a_1, a_2) \leq n$, as can be seen from the CNF transformation for monadic formulae.

The number of clauses given by the first scheme can be bounded by $c_1 := (2^{n_2} \cdot 2^{2n_1}) \cdot 2^{2n_1} = 2^{O(n)}$. The number of *irreducible* clauses given by the second scheme (that cannot be simplified, in particular by the splitting rule) can be bounded by $c_2 := 2^{a_1} \cdot 2^{2n_1} \cdot (2^{(2n_1)})^{a_1} = 2^{O(n^2)}$: for each of at most a_1 variables there can be at most $2n_1$ unary literals in the clause. The number of irreducible clauses given by the third scheme is bounded by $c_3 := 2^{n_2} \cdot c_2 = 2^{O(n^2)}$. Similarly, the number of irreducible clauses of the forth scheme $c_4 \leq c_3 = 2^{O(n^2)}$. So, the total number of relevant clauses from (M) is bounded by $c_m = 2^{O(n^2)}$. A smaller complexity bound $c_m = 2^{O(n)}$ can be obtained if we reconsider the structural transformation for monadic formulae. Note, that each unary definitional predicate can occur in a clause with the unique variable (exactly, for which it was introduced). Therefore, the maximal number of clauses given by the second scheme can be recomputed as $c_2 = 2^{a_1} \cdot 2^{2n_1} \cdot (2^{(2n_1)}) = 2^{O(n)}$. The bounds for the other cases can be lowered to $2^{O(n)}$ accordingly.

Appendix B
Proof of Theorem 7

In this appendix we give the details for the proof of Theorem 7.

$\boldsymbol{\mathcal{GF}|\mathcal{M}_f}$ **:** Given a formula $F \in \mathcal{GF}|\mathcal{M}_f$ of the size n and width w, ordered resolution nondeterministically produce a saturation that is a subset of (G)+(M). So, the number of relevant clauses for F is bounded by $c_{gm} := c_g + c_m = 2^{n^2 \cdot 2^{O(w \log w)}}$. On every saturation branch, the number of applications of the splitting rule is bounded by $c_{\mathtt{SP}} = 2^{O(n^2)}$ that is the number of different clauses of the form `M.2.3`. Therefore, the size of the saturation tree is bounded by $2^{c_{\mathtt{SP}} \cdot c_{gm}}$ and $\mathcal{GF}|\mathcal{M}_f$ is decided by a 2EXPTIME procedure (with backtracking).

$\boldsymbol{\mathcal{GF}^k|\mathcal{M}_f}$ **:** Let $F \in \mathcal{GF}^k|\mathcal{M}_f$ be of the size n and width w. Note that w can be greater than k, since the width of monadic formulas is not bounded by a constant. Therefore, the straightforward estimation of the number of relevant clauses gives us the same complexity as for $\mathcal{GF}|\mathcal{M}_f$. To obtain a better complexity, note that: ***(i)*** The arities of all definitional predicate symbols and functional symbols in the initial guarded clauses are not greater than k. ***(ii)*** A resolution inference between a guarded and a monadic clause is possible only if definitional predicates or functional terms are unified, therefore the property ***(i)*** also holds for the conclusion of the inference. ***(iii)*** All literals in monadic clause that do not contain definitional predicates are the instances of one-variable literals. Therefore, all relevant clauses from (G) contain only instances of initial literals with at most k free variables and the number of such a clauses is bounded by $c_{g^k} := 2^{n \cdot 2^{O(k \log k)}}$. Thus, the number of relevant clauses from (G)+(M) is bounded by $c_{g^k m} := c_{g^k} + c_m = 2^{O(n^2)}$, yielding the intended NEXPTIME upper bound for the procedure.

$\boldsymbol{\mathcal{FO}^2|\mathcal{M}_f}$ **:** The arguments that allow to bound the number c_t of relevant clauses in (T) are the same as were given for $\mathcal{GF}^k|\mathcal{M}_f$. So $c_t := 2^{O(n \cdot 2^k)}$ and the total number of relevant clauses from (T)+(M) is bounded by $c_{tm} := c_t + c_m = 2^{O(n^2)}$. This shows that the decision procedure for $\mathcal{FO}^2|\mathcal{M}_f$ can be implemented in NEXPTIME.

$\boldsymbol{\mathcal{GF}|\mathcal{FO}^2|\mathcal{M}_f}$ **:** The number of relevant clauses is estimated in the same way as for $\mathcal{GF}|\mathcal{M}_f$, which gives the 2EXPTIME upper bound for the procedure.

$\boldsymbol{\mathcal{GF}^k|\mathcal{FO}^2|\mathcal{M}_f}$ **:** The complexity is estimated in the same way as for $\mathcal{GF}^k|\mathcal{M}_f$ and $\mathcal{FO}^2|\mathcal{M}_f$, giving the intended NEXPTIME complexity bound.