HABILITATIONSSCHRIFT

Using Resolution as Decision Procedure

eingereicht an der Technischen Universität Wien

Fakultät für Informatik

von

Dr. Ir. Jean Marie Guillaume Gérard de Nivelle
Scheidterstrasse 62
66125 Dudweiler
Deutschland

Juni 2005

1 Introduction

Logic made a major step in 1879, when Gottlob Frege published his 'Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens', see [26]. In this paper, which marks the start of modern logic, Frege introduced a formal language, (which he called Begriffsschrift), in which mathematical statements can be formally stated, and also formally proven. (without any resort to intuition)

The reasons for the development of mathematical logic around 1900 were the problems that arose when mathematics moved attention towards truly abstract notions, like for example general functions, general sequences and sets. For those abstract mathematical structures, intuitive reasoning turned out problematic. (For example, Frege started his research on logic when he tried to analyze the notion of sequence)

In the beginning of the twentieth century, various systems had been developed as foundational instrument, see [26]. The largest work in this direction was the Principia Mathematica [21], in which properties of cardinals and ordinals were formalized. In later volumes, the integers, the rationals, and the reals were introduced, and some of their properties were formally proven.

Once one has a formal calculus in which mathematical proofs can be verified, the next natural question is: Is there a mechanical method for finding those proofs? Unfortunately, this question was answered with 'no' around 1930. There do exist algorithms that are guaranteed to find a proof if there exists one, but there exists no algorithm that is guaranteed to report an answer in the case that no proof exists. But still one can try to find an algorithm that searches for a proof and that is guaranteed to find a proof if there exists one. This became particularly interesting when computers became generally available, in the second half of the twentieth century.

For many applications, establishing that no proof exists is equally important as finding a proof, and an algorithm that can only stop on positive answers is not enough. Examples of such applications are question answering from a database, or anaphora resolution in natural language processing: A user asking for all flights from A to B also wants to receive an answer, when there is no such flight. In anaphora resolution, deduction can be used for deciding possible references of anaphora. Consider the discourses 'After the Titanic hit the iceberg, it sank', or 'After the Titanic hit the iceberg, it slowly melted'. In order to determine what 'it' refers to, one needs to be able to reason with background knowledge. If one knows that ships don't melt, and icebergs don't sink, then the reference of 'it' can be determined in both cases. In order to be of practical use, the system that reasons with the background knowledge needs to terminate.

For sets of formulas that can formalize integers, there exists no terminating proof search strategy. Fortunately for many applications, it is sufficient to consider weaker sets of formulas. For many of those, it turns out that there exist terminating algorithms. Often, one can also give a guarantee on the time needed to find or not-find a proof.

In the papers on which this habilitation thesis is based, we studied the use of resolution as a decision procedure. Resolution is a frequently used method for automated proof search, which can be used for full first-order logic. As a consequence, resolution does not terminate in general. We study modifications of resolution that can be used as decision procedures for subsets of first-order logic, and try to give upper bounds on the time/space of the proof search.

Resolution was introduced by Robinson in [20]. It makes use of a normal form for first-order logic, which is called *clausal normal form.* The advantage of the normal form is that, once the formula is transformed into this normal form, a relatively simple calculus, consisting of only a few rules can be used. The complete procedure takes the following form: First, the formula (or set of formulas) is transformed into clausal normal form. There are different ways to do this, which are optimized for different types of problems.

When the formula is in clausal normal form, it consists of a set of formulas of a very simple form, which is called *clause.* Using a few, relatively simple rules, consequences of these clauses are derived, which are also clauses. When a special clause, the *empty clause*, is derived, one knows that the original set of clauses was provable. Because the new clauses can in turn take part in derivations, there is no guarantee that this process will end.

Since its introduction, there have been many improvements of resolution, mostly with the purpose of improving resolution for general-purpose theorem proving. There have been improvements on the strategies for transforming a formule into CNF, see [2, 19]. Various restrictions of resolution have been developed that preserve completeness [22, 3]. In addition, deletion strategies have been developed: Under certain conditions, when a new clause is derived, which is logically implied by existing clauses, it can be deleted from the database, see [4].

Until now, resolution and its extensions to equality, are still the most successful technique for general first-order theorem proving, although for certain classes of problems other techniques, like semantic tableaux, or instantiation based methods are emerging. (See for example [5, 18, 12])

The topic of this habilitation thesis is the usage of resolution as a decision procedure. The possibility of using resolution as a decision procedure was first considered in [17]. Joyner showed that certain restrictions of resolution, which are closely related to those used for improving efficiency, can also be used to turn resolution into a decision procedure. The research on resolution decision procedures was continued by the authors of [11], who were able to give various decision procedures for natural subsets of first-order logic. This habilitation thesis builds strongly on [11]. In [6], an open problem of [11] was solved. In Section 2.4 of this thesis, another one is addressed.

2 Paper Overview

This cumulative habilitation thesis is based on five of my publications since 1999. The five papers are related to the use of resolution as decision procedure.

I will shortly introduce each of the five papers:

2.1 Deciding the Guarded Fragments by Resolution

In this paper resolution decision procedures for the guarded fragment and the loosely guarded fragment are given. The guarded fragment is a decidable fragment of first-order logic that was introduced by Johan van Benthem, Hainal Andréka, and István Németi in [1]. Their was to 'identify the modal fragment of first-order logic'. We first give the definition:

Definition 2.1 *A first-order formula is in the guarded fragment if*

1. *The formula contains no function symbols,*
2. *Every universal quantification has form $\forall \overline{x}\ A \rightarrow F$, where A is an atom, containing at least all free variables of A and F.*
3. *Every existential quantification has form $\exists \overline{x}\ A \wedge F$, where A is an atom, containing at least the free variables of A and F.*
4. *The formula contains no equality.*

The original definition of the guarded fragment contained no equality. Later it was shown in [14] that the guarded fragment remains decidable when equality is added, as long as equality is not used as a guard. In the next section, we give a resolution decision procedure for the guarded fragment with equality. The relevance of the guarded fragment lies in the fact that formulas from many modal logics (including e.g K and B) can be translated into it.

For example, the modal formula $\Box(A \wedge \Diamond B)$ can be translated into

$$\forall y\ (\ R(x,y) \rightarrow A(y) \wedge \exists z\ (\ R(y,z) \wedge B(z)\ )\ ).$$

By a carefully chosen clause transformation, formulas from the guarded fragment can be translated into clauses of the following form:

Definition 2.2 *A clause is guarded if*

1. *the clause is ground, or*
2. *it contains a negative, function free literal which contains all variables of the clause, and every non-ground, functional term in the clause contains all variables of the clause.*

In the second case, a negative literal which contains all variables is called guard *of the clause.*

First-order, guarded formulas can be translated into guarded clauses by an appropriately chosen clause transformation. First the formula has to be transformed into NNF as standard. After that, new predicates $\alpha(\overline{y})$ are introduced for subformulas of form $\forall \overline{x}\ A \rightarrow F$. Here $\overline{y}$ are the free variables of $\forall \overline{x}\ A \rightarrow F$.

Obviously, $\overline{y} \subseteq \overline{x}$. For the $\alpha(\overline{y})$, defining formulas of form $\forall \overline{x}\ A \rightarrow \alpha(\overline{y}) \rightarrow F$ are added.

In the paper, the following ordering refinement is introduced for the guarded fragment:

Definition 2.3 *We define the order $\prec$ on atoms from $A \prec B$ iff either*

1. *The variables of A are a strict subset of the variables of B, or*
2. *the deepest occurrence of a variable in A is strictly less deep than the deepest occurrence of a variable in B.*

The order $\prec$ is extended to literals in the usual way:

$$A \prec B \text{ implies } \neg A \prec B, \quad A \prec \neg B, \quad \neg A \prec \neg B.$$

The following example gives a few guarded and non-guarded clauses. In the guarded clauses, the literals that can be resolved upon are marked with a star.

Example 2.4

guarded clauses	*non-guarded clauses*
$\neg p(X,X,Y)^* \vee q(X) \vee p(Y,X)$,	$\neg p(f(X,Y)) \vee q(X,Y)$,
$\neg p(X,Y) \vee q(X,f(X,Y))^*$,	$\neg p(X,Y) \vee q(X,f(Y))$,
$\neg p(s(0),s(0))^* \vee p(0,0)^*$.	$\neg p(X,Y) \vee q(X,Y,Z)$.

It can be shown that the resolution strategy preserves the form of the clauses, and that the variable depth of the clauses does not increase. As a consequence, only a finite set of clauses can be derived from a given initial clause set. It is also shown that this strategy is complete for guarded clauses.

Although the guarded fragment contains the translations of some modal logics, there are also quite a few modal logics, which cannot be straightforwardly translated into the guarded fragment. Examples are temporal modal logics able to express that some formula P is true everywhere between two moments of time x and z, like in $\forall y\ B(x,y) \wedge B(y,z) \rightarrow P(y)$. Here $B(x,y)$ means that x is before y. In order to express such properties, Van Benthem introduced the *loosely guarded fragment* in [25] and showed that it is decidable.

Definition 2.5 *The* loosely guarded fragment *is the following fragment of first-order logic.*

(1) *The formula contains no function symbols.* **(2)** *Every universal quantification has form $\forall \overline{x}\ A_1 \wedge \cdots \wedge A_p \rightarrow F$, where $A_1 \wedge \cdots \wedge A_p$ is a conjunction of atoms, s.t. for every pair of variables α, β, for which $\alpha \in \overline{x}$ and β a free variable of in $A_1, \ldots, A_p, F$, it must be the case that α and β occur together in one of the A_i, $1 \leq i \leq p$.* **(3)** *Every existential quantification has form $\exists \overline{x}\ A_1 \wedge \cdots \wedge A_p \wedge F$, with the same conditions as under (3).* **(4)** *The formula contains no equality.*

The following clause fragment corresponds to the loosely guarded fragment:

Definition 2.6 *A clause is* loosely guarded *if* **(1)** *either it is ground, or* **(2a)** *it contains a set of negative literals* $\{\neg A_1, \ldots, \neg A_p\}$*, which are function free and each pair of variables occurring in the clause, occurs together in one of the* A_i*, and* **(2b)** *every non-ground, functional term contains all variables of the clause.*

Definition 2.6 should be read as $1 \vee (2a \wedge 2b)$.
In the paper, the loosely guarded fragment is decided by the following (admittedly complicated) selection strategy:

- Factoring is always permitted.
- In clauses of Form 1, only literals containing terms of maximal depth can be resolved upon.
- In clauses of Form 2 which contain non-ground functional terms, resolution is allowed only upon literals containing a non-ground functional term at maximal depth.
- Non-ground clauses without non-ground, functional terms can only be used in special, hyperresolution-like inferences: Write the clause in the form: $\{\neg A_1, \ldots, \neg A_p\} \cup R$, where $\{\neg A_1, \ldots, \neg A_p\}$ is a loose guard. If there are clauses
$$R_1 \cup [\ B_1\ ], \ldots, R_p \cup [\ B_p\ ],$$
which are ground or have non-ground, functional terms, in which the B_i can be resolved upon, and there exists a most general unifier Θ, such that $B_1\Theta = A_1\Theta, \ldots, B_p\Theta = B_p\Theta$, then construct a partition
$$[\ \neg A'_1, \ldots, \neg A'_{p'}\ ] \cup [\ \neg A''_1, \ldots, \neg A''_{p''}\ ] \text{ of } [\ \neg A_1, \ldots, \neg A_p\ ]$$
and the corresponding partition
$$\{\ R'_1 \cup [\ B'_1\ ], \ldots, R'_{p'} \cup [\ B'_{p'}\ ]\ \ \} \cup \{\ R''_1 \cup [\ B''_1\ ], \ldots, R''_{p''} \cup [\ B''_{p''}\ ]\ \ \} \text{ of}$$
$$\{\ R_1 \cup [\ B_1\ ], \ldots, R_p \cup [\ B_p\ ]\ \},$$
s.t. the partial hyperresolvent
$$R'_1\Sigma \cup \cdots \cup R'_{p'}\Sigma \cup [\ \neg A''_1\Sigma, \ldots, \neg A''_{p''}\Sigma\ ] \cup R\Sigma.$$
has no non-ground, functional term at a position that is deeper than the positions at which non-ground, functional terms occur in the premisses $R''_1 \cup [\ B''_1\ ], \ldots, R''_{p''} \cup [\ B''_{p''}\ ]$.

It is shown that this complicated strategy is complete and that it always terminates. In particular, it is proven that a partition as required by the last rule, always exists.

The original goal of [1] was more ambitious than identifying *some* fragment containing the translation of *some* modal logics. What the authors actually wanted was to characterize exactly *the modal fragment of first-order logic.* We come back to this in Section 2.3.

2.2 A Superposition Decision Procedure for the Guarded Fragment with Equality

The paper presents a modification of the strategy of the previous section, such that it can decide the guarded fragment with equality. The order $\prec$ of Definition 2.3 is non-liftable (not preserved under substitution). As a consequence, it cannot be combined with paramodulation without loosing completeness. However, the clause fragment of Definition 2.2 is larger than necessary, because in clauses originating from clausification, there are no nested function symbols. We showed that if one restricts the nesting depth to one, then one can use a liftable order and selection.

The guarded fragment with equality is obtained by dropping Condition 4 in Definition 2.1. Different from [14], we allow equality to act as guard. An example of a clause that is guarded by equality, is $\forall x\ (\ x \approx x \rightarrow p(x)\ )$. Because $x \approx x$ is a tautology, this is equivalent to allowing unguarded quantification over a single variable. Guarded clauses with equality can be translated into the following clause fragment:

Definition 2.7 *A clause c is a* guarded clause with equality *if it has one of the following three forms:*

1. *c is a ground clause and contains no nested function symbols,*
2. *c contains one variable and no function symbols, or*
3. *every functional subterm in c contains all variables of c and c contains a negative literal without function symbols, which contains all variables of c.*

Note that in the third case, it is not required that the clause contains more than one variable. A clause with one variable may contain function symbols, but then it has to contain a guard as well. It can be checked that Definition 2.7 is equivalent to Definition 4.2 in [13]. On one hand, Definition 2.7 is more general than Definition 2.2, because it allows equalities and unguarded, one-variable clauses. On the other hand, it does not allow nesting of functional symbols.

The decision strategy for guarded clause sets is an instance of superposition with selection. Superposition is a restriction of paramodulation, in which orders are used not only for determining which literals in a clause can be used, but also for determining in which direction equalities can be used. Selection makes it possible to overrule the ordering, as long as a negative literal is selected.

Definition 2.8 *A* simplification order *is a strict, total order $\prec$ on ground terms satisfying the following conditions:*

- *$\prec$ is well-founded.*
- *$\prec$ is preserved in contexts: If $t_1 \prec t_2$, then $T[t_1] \prec T[t_2]$.*

Definition 2.9 *A* selection function Σ *is a function from clauses to clauses, s.t. if for some clause c, $\Sigma(c)$ is not the empty clause, then all literals in $\Sigma(c)$ are negative, and $\Sigma(c) \subseteq c$.*
If $\Sigma(c) \neq [\ ]$, we say that Σ has selected *the literals in $\Sigma(c)$.*

When $\Sigma(c)$ is empty, the order determines which literals in the clause can be used. Otherwise, only the literals in $\Sigma(c)$ can be used. In [3], it was shown that superposition (combined with some other rules) is complete for every combination of a selection function Σ and simplification order $\prec$. In [13], it is shown that the following order and simplification order cause superposition to terminate on the guarded fragment with equality:

Definition 2.10 *Selection function Σ is defined as follows: If c is a guarded clause of Type 3, and c does not contain function symbols, then $\Sigma(c)$ contains the guards of c. Otherwise, $\Sigma(c)$ is empty.*

The order $\prec$ on terms and literals is defined as follows: For terms t_1, t_2 define $t_1 \prec t_2$ if t_2 contains function symbols, while t_1 does not. Similarly, for literals A, B, define $A \prec B$ if B contains function symbols, while A does not.

It is shown that standard orders (like KBO or RPO) can be tuned in such a way that they fullfil the conditions of Definition 2.10. The essential property of the strategy is that eligible literals and terms always contain all variables of the clause in which they occur.

It is also shown that the complexity of the decision procedure is optimal. Since the strategy preserves the clause format of Definition 2.7, the complexity of the procedure can be estimated by determining the number of guarded clauses that is possible over a given signature:

Theorem 2.11 *Assume a signature containing p predicate symbols (including equality) with maximal arity a_p, containing f function symbols with maximal arity a_f, and containing c constants.*
Then Definition 2.7 allows

$$c + a_p \text{ function free terms,}$$

$$c + a_p + f(c + a_p)^{a_f} \text{ terms,}$$

$$p(c + a_p + f(c + a_p)^{a_f})^{a_p} \text{ atoms,}$$

$$3^{(p(c+a_p+f(c+a_p)^{a_f})\ )^{a_p}} \text{ clauses.}$$

The last number is doubly exponential in the signature and corresponds to the lowerbound of [14].

2.3 Deciding Regular Grammar Logics with Converse through First-Order Logic

One of the original motivations for introducing the guarded fragment, according to [1], was to identify the modal fragment of first-order logic. Although not a precise statement, it probably means that modal logics should be easily translatable into it, that it should have a complexity comparable to the complexities of modal logics, and that it should have good metalogical properties, like modal logics have (as e.g. interpolation)

In [8], [9], we concentrate on one aspect, namely the decidability of modal logics by translation into the guarded fragment. Using the standard relational translation, modal formulas from logic K can be straightforwardly translated into the guarded fragment as follows:

Definition 2.12 *Let ψ be a (multi)modal formula. We assume two fixed variables x and y. The relational translation $t_{\mathrm{rel}}(\psi)$ of a modal formula ψ is defined as $\exists xy\ t_{\mathrm{rel}}(\psi, x, y)$, where $t_{\mathrm{rel}}/3$ in turn is recursively defined as follows:*

- $t_{\mathrm{rel}}(p, \alpha, \beta) = \mathbf{P}(\alpha)$,
- $t_{\mathrm{rel}}(\neg\psi, \alpha, \beta) = \neg\ t_{\mathrm{rel}}(\psi, \alpha, \beta)$,
- $t_{\mathrm{rel}}(\psi_1 \wedge \psi_2) = t_{\mathrm{rel}}(\psi_1, \alpha, \beta) \wedge t_{\mathrm{rel}}(\psi_2, \alpha, \beta)$,
- $t_{\mathrm{rel}}(\psi_1 \vee \psi_2) = t_{\mathrm{rel}}(\psi_1, \alpha, \beta) \vee t_{\mathrm{rel}}(\psi_2, \alpha, \beta)$,
- $t_{\mathrm{rel}}(\psi_1 \rightarrow \psi_2) = t_{\mathrm{rel}}(\psi_1, \alpha, \beta) \rightarrow t_{\mathrm{rel}}(\psi_2, \alpha, \beta)$,
- $t_{\mathrm{rel}}(\psi_1 \leftrightarrow \psi_2) = t_{\mathrm{rel}}(\psi_1, \alpha, \beta) \leftrightarrow t_{\mathrm{rel}}(\psi_2, \alpha, \beta)$,
- $t_{\mathrm{rel}}(\langle a\rangle\psi,\ \alpha, \beta) = \exists\beta\ [\ \mathbf{R}_a(\alpha, \beta) \wedge t_{\mathrm{rel}}(\psi, \beta, \alpha)\ ]$,
- $t_{\mathrm{rel}}([a]\psi,\ \alpha, \beta) = \forall\beta\ [\ \mathbf{R}_a(\alpha, \beta) \rightarrow t_{\mathrm{rel}}(\psi, \beta, \alpha)\ ]$.

It is easily seen that the relational translation translates modal formulas into the guarded fragment. In case one wants to translate a formula that belongs to a modal logic different from K, one has to add the frame property to the translation. For example, symmetry, which is required by modal logic B, has frame property $\forall xy\ R(x, y) \rightarrow R(y, x)$, which is in the guarded fragment.

Unfortunately, there are many modal logics which have a low complexity, but which have frame properties that cannot be translated into the guarded fragment. If one wants to test satisfiability of a modal formula ψ in the modal logic $S4$, then this has to be done by testing satisfiability of

$$t_{\mathrm{rel}}(\psi) \wedge \forall w\ R(w, w) \wedge \forall w_1 w_2 w_3\ [\ R(w_1, w_2) \wedge R(w_2, w_3) \rightarrow R(w_1, w_3)\ ].$$

The transitivity axiom is not in the guarded fragment. This can be shown by observing that transitivity is not preserved by guarded bisimulation.

There have been quite a few attempts to extend the guarded fragment, in order to be able to capture capture S4-like logics. For example, it has been shown by Szwast and Tendera [23] that the guarded fragment remains decidable when transitivity axioms are added, as long as the transitive predicates occur only in guards. In [15], it has been shown that it is possible to add monotone fixed point operators to the guarded fragment. Modal logic $S4$ can be translated into both fragments. Both fragments have the same complexity as the guarded fragment. In our view, however, the fragments do not really explain the low complexity of modal logic S4, because the principles used in the decidability proofs and the resulting algorithms are much more complex than those needed for $S4$. This is also reflected by the fact that there exist no implementations of the guarded fragment with fixed points, or the guarded fragment with transitive guards.

In [9] and [10], a different approach is chosen. Instead of adding the axioms and extending the fragment, one can slightly modify the translation, so that $S4$-formulas can be translated into the guarded fragment.

Definition 2.13 *A multimodal formula ψ is in* negation normal form *if it does not contain $\leftrightarrow$ and $\neg$ is applied only on atoms.*

Definition 2.14 *Let ψ be a multimodal formula in negation normal form. We define the relational translation for $S4$, $t_{S4}(\psi)$ as $\exists xy\ t_{S4}(\psi, x, y)$, where $t_{S4}/3$ in turn is recursively defined as follows:*

- $t_{S4}(p, \alpha, \beta) = \mathbf{P}(\alpha)$,
- $t_{S4}(\neg p, \alpha, \beta) = \neg\mathbf{P}(\alpha)$,
- $t_{S4}(\psi_1 \wedge \psi_2) = t_{S4}(\psi_1, \alpha, \beta) \wedge t_{S4}(\psi_2, \alpha, \beta)$,
- $t_{S4}(\psi_1 \vee \psi_2) = t_{S4}(\psi_1, \alpha, \beta) \vee t_{S4}(\psi_2, \alpha, \beta)$,
- $t_{S4}(\langle a\rangle\psi,\ \alpha, \beta) = \exists\beta\ [\ \mathbf{R}_a(\alpha, \beta) \wedge t_{S4}(\psi, \beta, \alpha)\ ]$,
- $t_{S4}([a]\psi,\ \alpha, \beta) =$
  $X(\alpha)\ \wedge$
  $\forall\alpha\beta\ [\ X(\alpha) \rightarrow \mathbf{R}_a(\alpha, \beta) \rightarrow X(\beta)\ ]\ \wedge$ *(X is a new symbol)*
  $\forall\alpha\ [\ X(\alpha) \rightarrow t_{S4}(\psi, \alpha, \beta)\ ]$.

It can be easily checked that for every multimodal formula ψ in negation normal form, the translation $t_{S4}(\psi)$ is within the guarded fragment. It can also be proven that ψ is satisfiable in $S4$ iff $t_{S4}(\psi)$ is satisfiable in first-order logic.

If $t_{S4}(\psi)$ has a first-order model, then one can replace the interpretations of the relations R_a by their transitive closures, and show that ψ is true in this model. Conversely, if $S4$ holds in a modal model in which the translations of the relations R_a are transitive, then one can extend it with interpretations for the X-predicates that satisfy the axioms in the translation.

In [9] and [10], a sufficient condition on the frame condition of the logic is given, which makes the translation possible. The characterization is based on formal language theory. In multimodal logic, the accessibility relations have form R_a, where a originates from some indexing set Σ. This set can be viewed as an alphabet. Then paths in modal models correspond to words over Σ. Given a closure condition, one can construct for each relation R_a, the set of paths which will be included in R_a by the closure. Using the language analogy, one can view, for each relation R_a, this set of paths as a formal language. The sufficient condition is that the translation is possible if for each $a \in \Sigma$, this 'path language' is regular.

In the case of $S4$, the logic is characterized by transitivity. With transitivy and reflexivity, the set of paths that will be included in R_a equals

$$\{\epsilon, a, aa, aaa, aaaa, \ldots\}.$$

This language is regular and can be recognized by an automaton with a single state X, which is initial and accepting state. The transition function consists of a single transition under a, which cycles in X. The automaton is reflected in the translation for $t_{S4}(\ [a]\psi,\ \alpha, \beta)$:

$$X(\alpha), \quad \forall\alpha\beta\ [\ X(\alpha) \rightarrow \mathbf{R}_a(\alpha,\beta) \rightarrow X(\beta)\ ], \quad \forall\alpha\ [\ X(\alpha) \rightarrow t_{S4}(\psi,\alpha,\beta)\ ].$$

In general, a frame condition can be viewed as an inclusion condition of form $R_{a_1} \cdot \ldots \cdot R_{a_n} \subseteq R_a$, where $a \in \Sigma$, and $a_1 \cdot \ldots \cdot a_n \in \Sigma^*$. Using the analogy with formal languages, the condition can be viewed as a grammar rule $a \rightarrow a_1 \cdot \ldots \cdot a_n$. This led us to the following definition:

Definition 2.15 *A multimodal logic is a* regular grammar logic *(with converse) if it can be expressed by a set of grammar rules $\mathcal{S}$, s.t. for every symbol a the set $\{w \mid a \Rightarrow_S^* w\}$ is regular.*

In [9] and [10], it is shown that for every regular grammar logic a translation similar to the translation in Definition 2.14 can be obtained. This makes it possible to obtain a decision procedure for regular grammar logics with converse. As a consequence, the satisfiability problem for regular grammar logics is in 2EXPTIME. For $S4$ this is worse than optimal, because $S4$ can be decided in PSPACE, but there are other regular grammar logics that are 2EXPTIME-hard, so for those, the translation is optimal.

A remaining question is the question of finding a necessary condition on a regular grammar logic under which a translation into the guarded fragment is possible. With such a condition, it would be possible to determine for a given grammar logic, whether an automaton-based translation into the guarded fragment exists. This problem is still under investigation.

2.4 Deciding the E^+-Class by an A Posteriori, Liftable Order

In this paper, a technical problem related to the E^+-class was solved. The E^+-class was introduced by Tanel Tammet in [24]. (It is also discussed on pages 108-109 of [11]) Unlike the other fragments discussed in this habilitation thesis, the E^+-class is defined as a clause class, and it does not correspond to a natural fragment of first-order logic. The E^+ class was obtained by generalizing the form of the clauses that originate from clausifying formulas in the one-variable fragment with function symbols. This fragment was shown decidable in [16]. Tammet called the clause fragment corresponding to the one-variable fragment E, which explains the name E^+ for the generalization. The problem that we solved was stated as an open question in [11], page 109. Before we can state it, some definitions are needed:

Definition 2.16 *A literal is* weakly covering *if every non-ground, functional term in the literal contains all variables that occur in the literal.*

A clause c is in E^+ if all its literals are weakly covering and have exactly the same set of variables.

A set of clauses is in E^+ if all its clauses are in E^+.

The definition of E^+ in [11] is slightly different from ours. There, for each pair of literals in a clause, the sets of variables have to be either identical, or disjoint. In case there are literals with disjoint sets of variables, the splitting rule can be applied, and after that the resulting fragments are the same.

Example 2.17 *The following literals are weakly covering:*

$$p(f(X,Y),X),\ q(f(s(0),X)),\ p(X,f(Y,g(X,Y,s(s(0))))),\ p(s(X),s(X)).$$

The following are not:

$$p(f(s(X),s(Y))),\quad p(f(X,Y),(X)).$$

The following clauses are in E^+ :

$$\{\ p(f(X,Y)),\ q(f(X,g(X,Y,s(0))))\ \},\quad \{\ \neg q(X),\ q(s(f(X,X)))\ \}.$$

The question that we solved is whether a certain order can be used to decide the E^+-class, when it is used a posteriori. We first define the order, then we define what it means to be applied a posteriori.

Definition 2.18 *We define the following order $\prec$ on literals: $A \prec B$ iff*

1. *The depth of A is strictly less than the depth of B,*
2. *and every variable that occurs in A has a deeper occurrence in B.*

The $\prec$-order is called $<_d$ in [11]. As usual, when counting depth, negation symbols are not counted.

Definition 2.19 A posteriori ordered resolution *and* a posteriori ordered factoring *are defined as follows:*

resolution: *Let $\{A_1\} \cup R_1$ and $\{\neg A_2\} \cup R_2$ be clausees which have no variables in common and s.t. A_1, A_2 have most general unifier Θ. If in addition, $A_1\Theta \not\prec R_1\Theta$ and $\neg A_2\Theta \not\prec R_2\Theta$, then one may construct the resolvent $R_1\Theta \cup R_2\Theta$.*

factoring: *Let $\{A_1, A_2\} \cup R$ be a clause, s.t. that A_1, A_2 have most general unifier Θ. If in addition, $A_1\Theta \not\prec [\ A_2\Theta\ ] \cup R\Theta$, then ony may construct the factor $\{A_1\Theta\} \cup R\Theta$.*

In order to see that a posteriori application of an order is stronger than a priori application, consider the following example:

Example 2.20 *Consider the following clause set, which is in E^+ :*

$$\{\ p(0, s^2(0))\ \},\quad \{\ \neg p(X, s^2(0)),\ p(s(X), s^2(0))\ \}.$$

Using $\prec$ a priori, the first literal of the second clause can be resolved upon, (because it is not strictly less deep than the second literal) and all clauses of form $[\ p(s^i(0), s^2(0))\ ]$, $i > 0$ are derivable. Using $\prec$ a posteriori, only the following clauses are derivable:

$$\{\ p(s(0), s^2(0))\ \},\quad \{\ \neg p(X, s^2(0)),\ p(s^2(X), s^2(0))\ \},\quad \{\ p(s^2(0), s^2(0))\ \}$$

In the example, resolution with a priori application of $<_d$ does not enforce termination, but resolution with a posteriori application of $<_d$ enforces termination. The main result of [7] is the following, which was posed as an open question in [11].

Theorem 2.21 *Let C be a set of clauses in E^+. Using a posteriori ordered resolution and factoring, the set of clauses that are derivable from C is finite.*

The relevance of this result lies in its contrast with the decision procedure using non-liftable orders, which was given in my thesis [6]. Here it is easy to prove completeness, because completeness of a posteriori ordered resolution with liftable orders is a standard result. Using a non-liftable order, termination is easy to prove, but then completeness is hard to prove.

2.5 A Resolution-Based Decision Procedure for the Two-Variable Fragment with Equality

The paper presents a decision procedure for the two-variable fragment with equality, which uses resolution, but in a more indirect way than the decision procedures in Sections 2.1, 2.2 and 2.4. The decision procedure contains an equality removal procedure that only works on formulas that are saturated under resolution. After the removal of equality, resolution has to be used one more time in order to obtain a decision procedure.

Definition 2.22 *A first-order formula is in the* two-variable fragment *if the formula contains no function symbols, and only two variables.*

The input of the procedure is a formula in the two-variable fragment of first-order logic, without function symbols, but possibly with equality. Without loss of generality, one may assume that the formula contains only binary and unary predicate symbols. In the first stage of the procedure, equality is removed. This is done by saturating certain parts of the formula under (a restricted form) of resolution. From the resulting formula, equality can be removed without changing satisfiability of the formula. The result is a formula, which is still in the two-variable fragment, but now without equality. It can be decided either by a non-liftable order, as was described in [6], or by indexed resolution, which is the procedure given in the paper.

We describe only the equality removal procedure, since that is the most interesting part of the paper. The first step of the procedure consists in transforming the formula into the followowing normal form:

Definition 2.23 *A formula A is in* conjunctive normal form *if it has form*

$$(L_{1,1} \vee \cdots \vee L_{1,l_1}) \wedge \cdots \wedge (L_{m,1} \vee \cdots \vee L_{m,l_m}),$$

with each $L_{i,j}$ a literal.

A formula F is in separation normal form *if it has form $F = \alpha \wedge \beta_1 \wedge \cdots \wedge \beta_n$, where α has form*

$$\forall xy\ (x \approx y \vee A),$$

with A quantifier free, equality free, in conjunction normal form, and containing no other variables than x, y.
Each β_i has one of the following two forms:

$$\exists x\ A_i \ \text{or}\ \forall x \exists y\ (x \not\approx y \wedge A_i),$$

with each A_i quantifier free, equality free, and in conjunction normal form. In the first case, A_i contains no other variable than x. In the second case, A_i contains no other variables than x and y.

Note that the term 'separation normal form' is not used in the paper. The formula $\forall xy\ (x \approx y \vee A)$ is equivalent to $\forall xy\ (x \not\approx y \rightarrow A)$. Every two-variable formula can be transformed into separation normal form by standard transformations. A subformula of form $\forall x \exists y\ A(x,y)$ can be replaced by $\forall x\ [A(x,x) \vee \exists y\ (x \not\approx y \wedge A(x,y)\ )\ ]$. Similarly, a subformula of form $\forall xy\ A(x,y)$ can be replaced by $\forall xy\ (x \approx y \vee A(x,y)\ ) \wedge \forall x\ A(x,x)$.

If one has a formula $\forall xy\ A$ with A in conjunctive normal form, then A can be viewed as a set of clauses, written in first-order notation, and one can perform resolution on it. Similarly, in $\forall x \exists y (x \not\approx y \wedge A_i)$, one can view A_i like a set of clauses.

Definition 2.24 *On a formula F in separation normal form, we allow the following restricted form of resolution: The literals resolved upon must be binary, and contain two distinct variables.*

We allow this form of resolution at two places: First between two disjunctions inside $\forall xy\ (x \approx y \vee A)$. The resolvent is added to this formula. Second between a disjunction in $\forall xy\ (x \approx y) \vee A$ and a disjunction in a $\forall x \exists y (x \not\approx y \wedge A_i)$. The resolvent has to be added to the A_i of the second formula.

It can be checked that this form of resolution is sound, and that it terminates in finite time. After that, the procedure continues as follows:

Theorem 2.25 *Let $F = \alpha \wedge \beta_1 \wedge \cdots \wedge \beta_n$ be a formula in separation normal form, that is closed under the resolution rules of Definition 2.24. Let $F' = \alpha' \wedge \beta_1 \wedge \cdots \wedge \beta_n$ be obtained from F by removing all disjunctions containing a two-variable literal from α. Then F has a model if and only if F' has a model.*

In the resulting formula, after the disjunctions with a two-variable literal have been removed, the only positive equality occurs in α' which has form $\forall xy\ (x \approx y \vee A')$. We know that A' contains only unary literals, or binary literals with a repeated variable. Hence we can write

$$A' = (\ C_1(x) \vee D_1(y)\ ) \wedge \cdots \wedge (\ C_m(x) \vee D_m(y)\ ),$$

where each C_j is a disjunction of unary atoms containing no other variable than x, and each D_j is a disjunction of unary atoms containing no other variable than y. The equality can be rearranged as

$$\alpha' = \forall xy\ (x \approx y \vee C_1(x) \vee D_1(y)\ ) \wedge \cdots \wedge \forall xy\ (x \approx y \vee C_m(x) \vee D_m(y)\ ).$$

Introducing new constants $e_1, \ldots, e_m$, this formula can be replaced by

$$\alpha'' = \forall x\ C_1(x) \vee \forall y\ D_1(y) \vee \forall x\ (\ C_1(x) \vee x \approx e_1\ ) \wedge \forall y\ (\ D_1(y) \vee y \approx e_1\ )$$

$$\cdots$$

$$\forall x\ C_m(x) \vee \forall y\ D_m(y) \vee \forall x\ (\ C_m(x) \vee x \approx e_m\ ) \wedge \forall y\ (\ D_m(y) \vee y \approx e_m\ ).$$

At this point, one can introduce unary predicates $E_j(x)$ for $x \approx e_j$. Uniformness of elements for which $E_j(x)$ holds can be ensured by the following axioms, which are all in the 2-variable fragment: For a unary predicate $P(\ )$ (including the $E_j(\ )$ predicates), one has $\forall xy\ (\ E_j(x) \wedge E_j(y) \wedge P(x) \rightarrow P(y)\ )$. For a binary predicate $P(\ ,\ )$, one has

$$\forall xy\ (\ E_j(x) \wedge P(x,y) \rightarrow \forall x_1\ (\ E_j(x_1) \rightarrow P(x_1,y)\ )),$$

and

$$\forall xy\ (\ E_j(y) \wedge P(x,y) \rightarrow \forall y_1\ (\ E_j(y_1) \rightarrow P(x,y_1)\ )).$$

In the resulting formula, there are only negative equalities left. For those, it suffices to add the axiom $\forall xy\ x \not\approx y$.

The result is a formula that is still within the two-variable fragment, but which does not contain equality. It can be decided by non-liftable order (see [6]), or by the variant of indexed resolution described in the paper.

3 Acknowledgements

I thank Prof. Alexander Leitsch for assisting me during the preparation of this habilitation thesis. I thank my colleagues from the Max-Planck Institut für Informatik, and the colleagues from the Institut für Computersprachen in Vienna for many pleasant visits. I thank my wife, Alexandra Kireeva for her unconditional support.
Finally, I want to thank B.C. Sliggers. No person had more influence on my scientific development.

References

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

[2] Matthias Baaz, Uwe Egly, and Alexander Leitsch. Normal form transformations. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 5, pages 275–333. Elsevier Science B.V., 2001.

[3] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[4] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–100. Elsevier Science B.V., 2001.

[5] Peter Baumgartner. A first-order Davis-Putnam-Logeman-Loveland procedure. In D. McAllester, editor, *CADE-17, 17th international conference on automated deduction*, volume 1831 of *LNAI*, pages 200–219. Springer Verlag, 2000.

[6] Hans de Nivelle. *Ordering Refinements of Resolution.* PhD thesis, Technische Universiteit Delft, 1995.

[7] Hans de Nivelle. Deciding the E^+-class by an a posteriori, liftable order. *Annals of Pure and Applied Logic*, 88(1):219–232, 2000.

[8] Hans de Nivelle and Stéphane Demri. Deciding regular grammar logics with converse through first-order logic. *Journal of Logic, Language and Information*, ?(accepted, expected in July):41 pages, July 2005.

[9] Stéphane Demri and Hans de Nivelle. Deciding regular grammar logics with converse through first-order logic. Research Report LSV-03-4, Laboratoire Spécification et Vérification, ENS Cachan, Cachan, France, February 2003. 29 pages, can be retrieved from `www.lsv.ens-cachan.fr/Publis/publis-y3-all.php`.

[10] Stéphane Demri and Hans de Nivelle. Deciding regular grammar logics with converse through first-order logic. arXiv:cs.LO/0306117, June 2003. Submitted to TOCL, can be retrieved from `arxiv.org/abs/cs.LO/0306117`.

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

[12] H. Ganzinger and K. Korovin. New directions in instantiation-based theorem proving. In *Proceedings 18th IEEE Symposium on Logic in Computer Science,(LICS'03)*, pages 55–64. IEEE Computer Society Press, 2003.

[13] Harald Ganzinger and Hans de Nivelle. A superposition decision procedure for the guarded fragment with equality. In Giuseppe Longo, editor, *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science (LICS-99)*, pages 295–303, Trento, Italy, 1999. IEEE Computer Society, IEEE Computer Society.

[14] E. Grädel. On the restraining power of guards. To appear in the Journal of Symbolic Logic, 1997.

[15] E. Grädel and I. Walukiewicz. Guarded Fixed Point Logic. In *LICS'99, Trento*, pages 45–54, 1999.

[16] Y. Gurevich. Formuly s odnim $\forall$ formulas with one $\forall$). In *Selected Questions in Algebra and Logics in memory of Mal'chev*, pages 97–110. Nauka, 1973.

[17] W.H. Joyner. Resolution strategies as decision procedures. *Journal of the ACM*, 23:398–417, 1976.

[18] R. Letz and G. Stenz. Proof and model generation with disconnection tableaux. In A. Voronkov, editor, *Proceedings of LPAR 2001*, volume 2250 of *LNAI*, pages 142–156. Springer Verlag, 2000.

[19] Andreas Nonnengart and Christoph Weidenbach. Computing small clause normal forms. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science B.V., 2001.

[20] J. A. Robinson. A machine oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[21] B. Russel and A.N. Whitehead. *Principia Mathematica.* Cambridge University Press, 1910-1927.

[22] J.R. Slagle. Automated theorem-proving for theories with simplifiers, commutativity and associativity. *Journal of the ACM*, 21(4):622–642, 1974.

[23] W. Szwast and L. Tendera. On the decision problem for the guarded fragment with transitivity. In *Proceedings of the 16th IEEE Symposium on Logic in Computer Science*, pages 147–156, 2001.

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

[25] J. van Benthem. Dynamic bits and pieces. Technical Report LP-97-01, Institute for Logic, Languard and Computation, University of Amsterdam, 1997.

[26] Jean van Heijenoort. *From Frege to Gödel, A source book in mathematical logic, 1879-1931.* Source books in the History of Sciences. Harvard University Press, 1967.

4 Selected Publications

Pages 20-56. Hans de Nivelle and Maarten de Rijke, Deciding the Guarded Fragments by Resolution, Journal of Symbolic Computation 35(1), pages 21-58, 2003.

Pages 57-65. Harald Ganzinger and Hans de Nivelle, A Superposition Decision Procedure for the Guarded Fragment with Equality, Proceedings of the 14th Annual IEEE Symposion on Logic in Computer Science (LICS-99), pages 295-303, 1999.

Pages 66-110. Hans de Nivelle and Stéphane Demri, Deciding Regular Grammar Logics with Converse through First-Order Logic, accepted for the Journal of Logic, Language and Information, expected appearance: July 2005.

Pages 111-128. Hans de Nivelle, Deciding the E^+-Class by an a Posteriori, Liftable Order, Annals of Pure and Applied Logic, 88(1), pages 219-232, 2000.

Pages 129-143. Hans de Nivelle and Ian Pratt-Hartmann, A Resolution-Based Decision Procedure for the Two-Variable Fragment with Equality, Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR-2001), LNCS 2250, pages 211-225, 2001.

J. Symbolic Computation (2005) **11**, 1–000

Deciding the Guarded Fragments by Resolution

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

(Received 22 June 2005)

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

1. Introduction

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

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

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

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

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

fragment and the loosely guarded fragment without equality. Recently, a superposition decision procedure for the guarded fragment with equality has been developed in (Ganzinger and de Nivelle, 1999). Although the first-order fragment in that paper is more general, the clause fragment had to be strongly restricted in order to make it possible to include equality. For example, the clause fragment used here allows nesting of function symbols, while this is not allowed in the other clause fragment. This makes that the decidability results here and the decidability results in (Ganzinger and de Nivelle, 1999) are incomparable at the clause level.
In order to decide the guarded fragment, we define guarded clauses, and show that first-order guarded formulae can be translated into sets of guarded clauses. After that we show that sets of guarded clause sets are decidable by an appropriate restriction of resolution. The restriction that has to be used is based on a so-called *ordering refinement.* All of the resolution theorem provers (SPASS (Weidenbach, 1997), OTTER (McCune, 1995), and Bliksem (de Nivelle, 1999a)) support orderings. This makes that our strategy fits very well into the standard framework of first-order resolution theorem proving. The standard optimizations and implementation techniques can be reused for our decision procedure, so we can expect our procedure to be technically efficient. Indeed, with an effective resolution-based decision procedure, implementation has become feasible. The strategy for the guarded fragment has been implemented in the theorem prover Bliksem (de Nivelle, 1999a). We will also show that our decision procedure is theoretically optimal, because it terminates in double exponential time.
In order to decide the loosely guarded fragment we define a similar notion of loosely guarded clause. However, deciding sets of loosely guarded clauses is much harder than deciding sets of guarded clauses. We need a non-trivial modification of hyperresolution on top of the ordering refinement for this. In order to prove its completeness, an extension of the resolution game turns out to be necessary.

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

2. Background

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

2.1. The Guarded Fragment

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

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

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

The atoms a in Item 4 are called guards.

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

Example 2.2. *The following formulae are guarded:*

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

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

The following formulae are not guarded:

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

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

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

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

2.2. Resolution

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

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

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

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

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

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

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

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

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

> *1 $\{A_1\} \cup R_1$ and $\{\neg A_2\} \cup R_2$ have no variables in common;*
> *2 there is no $A \in R_1$ such that $A_1 \sqsubset A$;*
> *3 there is no $A \in R_2$ such that $A_2 \sqsubset A$; and*
> *4 A_1 and A_2 have an mgu Θ.*
>
> *Then the clause $R_1\Theta \cup R_2\Theta$ is called a $\sqsubset$-*ordered resolvent *of $\{A_1\} \cup R_1$ and $\{\neg A_2\} \cup R_2$.*

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

> *1 there is no $A \in R$ such that $A_1 \sqsubset A$*;
> *2 A_1 and A_2 have an mgu Θ.*
>
> *Then the clause $\{A_1\Theta\} \cup R\Theta$ is called a $\sqsubset$-*ordered factor *of $\{A_1, A_2\} \cup R$.*

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

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

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

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

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

Then the clause

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

is a hyperresolvent.

2.3. Transformation to Clausal Normal Form

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

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

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

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

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

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

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

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

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

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

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

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

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

The result of Cls is of a set of clauses.

2.4. Weakly Covering Literals

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

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

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

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

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

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

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

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

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

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

2.5. The Resolution Game

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

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

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

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

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

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

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

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

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

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

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

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

3. The Guarded Fragment

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

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

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

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

3.1. Basics

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

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

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

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

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

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

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

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

Example 3.4. The guarded formula

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

is translated as follows. First, NNF results in

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

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

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

Sk results in

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

And finally, clausification results in

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

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

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

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

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

3.2. Termination

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

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

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

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

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

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

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

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

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

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

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

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

Claim 2 Condition 2 is preserved.

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

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

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

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

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

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

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

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

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

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

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

2 The case where c is obtained by factoring is analogeous.
$\Box$

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

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

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

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

3.3. COMPLETENESS

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

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

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

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

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

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

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

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

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

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

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

PROOF. One can delete the ground instance from every derivable representation-indexed clause, and show that it is still derivable. $\Box$

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

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

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

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

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

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

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

equals the indexed clause

$$\{a_1{:}(k, \overline{A_1}), \ldots, a_p{:}(k, \overline{A_p})\}.$$

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

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

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

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

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

We are now ready for the completeness proof.

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

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

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

be a list of substitutions such that the set of clauses

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

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

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

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

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

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

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

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

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

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

Then we can write

$$c_1 = \{a{:}(k_1, \overline{A}_1)\} \cup r_1{:}(k_1, R_1) \text{ and } c_2 = \{\neg a{:}(k_2, \overline{A}_2)\} \cup r_2{:}(k_2, R_2).$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

□

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

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

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

3.4. COMPLEXITY

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

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

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

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

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

and can be obtained in time

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

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

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

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

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

So we get

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

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

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

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

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

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

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

Then the set of possible clauses has at most size

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

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

4. The Loosely Guarded Fragment

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

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

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

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

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

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

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

Example 4.2. The transitivity axiom

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

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

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

4.1. Translation to CNF

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

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

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

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

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

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

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

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

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

Then the disjunction

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

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

4.2. Termination

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

Example 4.5. The following clause is loosely guarded:

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

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

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

using the substitution

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

The result equals

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

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

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

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

The result equals:

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

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

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

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

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

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

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

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

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

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

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

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

and

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

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

1 $\mathrm{Varnr}(B'_1\Theta') \leq \mathrm{Varnr}(B'_1)$, *and* $\mathrm{Vardepth}(B'_1\Theta') \leq \mathrm{Vardepth}(B'_1)$.
2 *For all i, with $1 \leq i \leq m$,*

$$\mathrm{Var}(B'_i\Theta') \subseteq \mathrm{Var}(B'_1\Theta'), \text{ and } \mathrm{Vardepth}(B'_i\Theta') \leq \mathrm{Vardepth}(B'_1\Theta').$$

3 *For all i, with $1 \leq i \leq m$,*

$$\mathrm{Var}(A'_i\Theta') = \mathrm{Var}(B'_i\Theta'), \text{ and } \mathrm{Vardepth}(A'_i\Theta') = \mathrm{Vardepth}(B'_i\Theta').$$

4 *For all i, with $1 \leq i \leq m$, both $A'_i\Theta'$ and $B'_i\Theta'$ are weakly covering.*

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

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

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

Then the following property holds:

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

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

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

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

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

and

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

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

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

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

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

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

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

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

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

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

Let Θ' be the mgu of the pairs

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

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

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

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

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

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

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

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

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

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

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

and

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

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

The result follows by collecting all the inclusions and inequalities. $\Box$

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

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

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

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

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

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

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

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

such that either

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

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

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

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

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

and construct a partial hyperresolvent as follows: From

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

and

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

construct

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

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

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

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

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

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

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

3 Let

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

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

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

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

$$\ldots$$

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

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

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

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

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

This implies that

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

From Theorem 4.7, Part 2, we have

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

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

□

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

4.3. Completeness

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

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

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

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

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

Having the selection function, when we construct the resolvent

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

we impose that condition that

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

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

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

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

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

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

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

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

The modified resolution game has the following derivation rules:

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

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

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

PARTIAL *Let*

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

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

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

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

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

(We have omitted the permutation for notational reasons)

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

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

We have the following completeness theorem:

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

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

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

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

We first establish the following property:

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

   1 The right hand side of c is selected,

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

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

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

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

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

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

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

In the induction proof we need the following property:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

be a clause in $[\overline{C}_{hb}]$. If there is an indexed literal $b_j\colon (k, \overline{B_j})$ containing non-ground, functional terms, then select the right hand side of c. Otherwise select the left hand side. We must show that when the right hand side is selected, the clause satisfies the condition

in Definition 4.13. Because the $a_i{:}(k, A_i)$ are part of the loose guard, they do not contain non-ground, functional terms. Let Σ be the substitution such that $a_i = A_i\Sigma$. Because A_i does not contain non-ground, functional terms, there exist no term A' and Σ', such that $a_i = A'\Sigma'$, and $\#\Sigma < \#\Sigma'$, so we know that $\#\Sigma \geq \#\Sigma'$. We also have $\#\Sigma \leq k$. (They are not necessarily equal because A_i need not contain all variables in the clause) From this it follows that there are no indexed atoms $a_i{:}(l, A_i') \in [\overline{C}_{hb}]$ with $k < l$, or $k = l$, and A_i' contains non-ground, functional terms.
We also need to show that for every atom occurring in a guard, there is a maximal indexed atom, based on a in $\mathcal{PA}$. This is the case because $\mathcal{PA}$ is finite.
It remains to show that $[\overline{C}_{hb}]$ is a saturation of the resolution game. This is essentially analogous to the proof of Theorem 3.19. The differences are the following:
When, due to substitution, a literal moves from the loose guard to the body of a clause, this is modelled by the first type of reduction, in Definition 4.13.
When a partial hyperresolvent is formed, assume that $[a_1, \ldots, a_p] \vdash r{:}(k, R)$ and

$$g_1 \vdash [a_1{:}(k_1, A_1)] \cup r_1{:}(k_1, R_1),$$

$$\ldots$$

$$g_p \vdash [a_p{:}(k_p, A_p)] \cup r_p{:}(k_p, R_p)$$

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

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

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

$$\ldots$$

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

$$\ldots$$

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

with partial hyperresolvent $h =$

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

$$\cup r{:}R\Theta \cup r_1{:}R_1\Theta \cup \cdots \cup r_m{:}R_m\Theta.$$

Write $[h] =$

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

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

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

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

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

□

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

5. Conclusions and Further Work

We have shown that it is possible to effectively decide the guarded fragment and the loosely guarded fragments by resolution. The proofs that the resolution refinements are complete and terminating can be used as proofs for the decidability of these fragments, but they offer more than that. They also define practical decision procedures, using techniques that are standard to the theorem proving community. This has made implementation relatively easy. Since the procedures could be built on top of an existing resolution prover, they could easily be combined with an efficient, full first order theorem prover (de Nivelle, 1999a)
Our decision procedure has interest in itself, but it can also be applied to modal logics, using the relational translation. From the space point of view, translation into the guarded fragment is not the optimal way for deciding simple modal logics like K and T, since these logics are in PSPACE (Ladner, 1977), while the complexity of the guarded fragment with fixed arity is single exponential. However it is not likely that a resolution decision procedure will ever decide modal logics in PSPACE, since resolution cannot even solve propositional logic in PSPACE.
We expect that our methods has advantages over the direct approaches of resolution in modal logic (Enjalbert and Fariñas del Cerro, 1989; de Nivelle, 1993), because our method provides a decision procedure, and because it can exploit existing implementations.
We do not expect to be able to improve the functional translation methods (Schmidt, 1997), at least not with our present translation.
A natural question is, whether or not the results in (Grädel and Walukiewicz, 1999) can be obtained by resolution. We are pessimistic but we will investigate the question.

Acknowledgments.

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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. It has been shown by Grädel (1997) 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 of first-order logic, 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 required, 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

[*]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

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 closed under boolean combinations.
4. If $F \in$ GF and G is an atom, for which every free variable of F is among the arguments of G, then $\forall\overline{x}(G \rightarrow F) \in$ GF (or, equivalently, $\forall\overline{x}(\neg G \vee F) \in$ GF) and $\exists\overline{x}(G \wedge F) \in$ GF, for every sequence $\overline{x}$ of variables.

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:

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

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:

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

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 Σ for negative literals. 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 $p(t_1, \ldots, t_k) \approx \mathtt{tt}$. In this encoding, the atom is considered a term (in a two-sorted signature with sorts i and o), with $\mathtt{tt}$ 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 $\mathtt{tt}$ is minimal. The multiset extension of $\succ$, again denoted $\succ$, is used to compare literals by identifying any positive equation $s \approx t$ (including the equational encodings of non-equational atoms) with the multiset $\{s, t\}$, and any negative equation $s \not\approx t$ with the multiset $\{s, t, \mathtt{tt}\}$, respectively. 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 and well-founded 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 non-deterministic 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 achieving good practical performance they have to be implemented. 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. We call a set of clauses N *saturated up to redundancy* (with respect to ordered paramodulation) if any inference from non-redundant premises in N is redundant in N. The definition of redundancy, in particular, implies that an inference is redundant in N if the conclusion of the inference is contained in N or else is redundant in N.

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 normal-form 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 sub-formulae by fresh names, together with a definition of the name.[1] We abstract universally quantified sub-formulae 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 sub-formula 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 sub-formula 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. This method of eliminating embedded quantifiers is standard and has also been used in the context of the guarded fragment by Grädel (1997).

For the purposes of this paper, the standard skolemization technique is the one which is appropriate. One 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 been removed, and Skolem function applications contain all the variables of a formula. 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:

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

$n(c)$
$\alpha(c)$
$\neg a(x,y) \vee \neg\alpha(x) \vee \beta(x)$
$\neg p(x,z) \vee \neg\beta(x) \vee a(f(x,z),z)$
$\neg p(x,z) \vee \neg\beta(x) \vee \neg b(z,z) \vee \neg c(fxz, fxz)$.

4.2 Guarded Clauses

The result of the transformation are sets of guarded clauses which, in particular, consist of a specific kind of literals. A term is called *shallow* if either it is a variable or else a functional term $f(u_1, \dots, u_m)$, $m \geq 0$, in which each u_j is a variable or a constant. A literal L is called *simple* if each term in L is shallow. 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 each 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 a 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 are shallow. Also, any literal in a guarded clause is covering.

These are some examples of guarded clauses where suitable guards have been underlined.
$p(0, s(0)) \vee c \not\approx d \vee q(s(0), f(0,0))$
$p(x,x) \vee q(x)$
$\neg p(y,x) \vee \underline{\neg q(x,y,y)} \vee r(x+y, x-y, x)$
$\underline{\neg p(y,x)} \vee \underline{\neg q(x,y,y)}$
$\underline{x \not\approx y} \vee x \approx (x+y)$
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)

Definition 4.2 is more restrictive than the corresponding definition in (de Nivelle 1998). The last two clauses in the previous example are guarded in the sense of (de Nivelle 1998). In the section 5 we will discuss this issue in more detail.

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 in the size of the signature.

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	$a_2 +$ the number of constant and function symbols
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 (modulo variable renaming) that may appear in a guarded clause over the given signature is bounded 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 bounded 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 $f \succ c \succ p \succ \mathtt{tt}$ 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 clause does not contain any constants. Let L_1 be a literal, and let t be a 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. Then, any of the u_i also occurs in t. With regard to the ordering, non-equational literals such as L_1 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 preceding 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 $D = D_1\sigma \vee D_2\sigma$. Notice that with $A = A_i$ and B any literal in D_i, the premises of the lemma 4.7 are satisfied, both for $i = 1$ and $i = 2$. 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. In case that C_1 is a positive, single-variable clause, then D contains at most one variable. If there is no guard in D then the resolvent must be a single-variable, positive, possibly empty 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 its 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 or 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. $\square$

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. As each inference rule is a decidable relation on guarded clauses, the theorem follows.[2] $\square$

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 which is not shallow by a new constant a_s, together with the defining equation $a_s \approx s$. This preserves satisfiability and produces a clause set which is guarded.

4.4 Complexity

The complexity of our decision procedure is double exponential. Grädel (1997) has shown that the decision problem for the guarded fragment with equality is 2EXPTIME-complete, hence our procedure is theoretically optimal. We use the fact, cf. Theorem 4.3, that the number of guarded clauses has a doubly exponential 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 (in the size of the signature).

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 complexity of $s * l * c$. For the time complexity, observe that suitable abstractions of the ordering and selection constraints 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 negligible 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. $\square$

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.

[2] The inferences are equipped with constraints which specify which literals are eligible for an inference. Depending on the signature, the term ordering, and the selection function such constraints are in general undecidable and have to be approximated. This is not the case here. But even if the constraints were undecidable, by Theorem 4.13 a safe approximation would be to consider any unrestricted inference the conclusion of which is a guarded clause.

THEOREM 5.2 Satisfiability 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 projection functions defined by equations of the form $f(x,y) \approx x$ can make a non-shallow term equal to a term that violates the covering condition. For example from the guarded clauses $\neg p(x,y) \ \vee \ p(s(f(x,y)))$ and $\neg p(x,y) \ \vee \ f(x,y) \approx x$ we may deduce the non-guarded clause $\neg p(x,y) \vee p(s(x))\}$, where s is not applied to all the variables of the clause. 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*. This fragment 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 $\neg A_1, \ldots, \neg A_n$, $n \geq 0$, called a *(loose) 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 Hyper-Resolution 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 hyper-resolution 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. 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:

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

From these one may derive the hyper-resolvent

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

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

A remedy to this problem is to resolve D only 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 its 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 a 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 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 of p, f, and g. If we disregard 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 if x is in $\mathrm{Var}(B_1, \ldots, B_n)$, either $x\sigma$ is of maximal depth d among all the $y\sigma$, for y in $\mathrm{Var}(B_1, \ldots, B_n)$, or else $x\sigma$ is a subterm of some $y\sigma$, with y in $\mathrm{Var}(B_1, \ldots, B_n)$. Picking for the j_i those atoms in which a variable x with $x\sigma$ of depth d appears, solves the problem. Any other variable in one of the B_{j_i} will be instantiated either by a term of the same depth and containing the same variables, or else by a direct subterm of a term of depth d.

The lemma covers exactly those unification problems which arise from hyper-resolution 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 partial inferences produces a (partial) conclusion which is a guarded clause.

The existence of suitable partial inferences solves our problem as the calculus remains complete if, for any potential hyper-inference from side premises $C_1, \ldots, C_k$ and main premise D, rather than deriving the full conclusion, we derive 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 proof does not dependent on any properties that are critical when adding equality. 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, simpler 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: The point here is that the loose guard of any generated clause is an instantiation of the loose guard of one of the parent clauses. Therefore, the number of variables in any derived clause is bound by the number of variables in one of the parent clauses.

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, Research Report ML-96-03, 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., 'Proceedings 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, 'Proceedings of Third Kurt Gödel Colloquium, KGC'93', Vol. 713 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 83–96. Revised version of Research Report MPI-I-93-204.

de Nivelle, H. (1998), A resolution decision procedure for the guarded fragment, *in* C. Kirchner & H. Kirchner, eds, 'Proceedings of the 15th International Conference on Automated Deduction (CADE-15)', Vol. 1421 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, Lindau, Germany, 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* 'Proceedings 14th IEEE Symposium on Logic in Computer Science', IEEE Computer Society Press, pp. 24–34.

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', *Journal of the 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.

Deciding regular grammar logics with converse through first-order logic

Stéphane Demri
LSV/CNRS UMR 8643 & INRIA Futurs projet SECSI & ENS Cachan
61, av. Pdt. Wilson, 94235 Cachan Cedex, France
email: demri@lsv.ens-cachan.fr

Hans De Nivelle
Max Planck Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
email: nivelle@mpi-sb.mpg.de

2000/04/29

Abstract. We provide a simple translation of the satisfiability problem for regular grammar logics with converse into GF^2, which is the intersection of the guarded fragment and the 2-variable fragment of first-order logic. The translation is theoretically interesting because it translates modal logics with certain frame conditions into first-order logic, without explicitly expressing the frame conditions. It is practically relevant because it makes it possible to use a decision procedure for the guarded fragment in order to decide regular grammar logics with converse. The class of regular grammar logics includes numerous logics from various application domains.

A consequence of the translation is that the general satisfiability problem for every regular grammar logics with converse is in EXPTIME. This extends a previous result of the first author for grammar logics without converse. Other logics that can be translated into GF^2 include nominal tense logics and intuitionistic logic. In our view, the results in this paper show that the natural first-order fragment corresponding to regular grammar logics is simply GF^2 without extra machinery such as fixed point-operators.

Keywords: modal and temporal logics, relational translation, guarded fragment, 2-variable fragment

1. Introduction

Translating modal logics. Modal logics are used in many areas of computer science, as for example knowledge representation, model-checking, and temporal reasoning. For *theorem proving* in modal logics, two main approaches can be distinguished. The first approach is to develop a theorem prover directly for the logic under consideration. The second approach is to translate the logic into some general logic, usually first-order logic. The first approach has the advantage that a specialized algorithm can make use of specific properties of the logic under consideration, enabling optimizations that would not work in general. Such optimizations often lead to terminating algorithms. In addition, implementation of a single modal logic is usually easier than implementing full first-order logic. But on the other hand, there are many modal logics, and it is simply not feasible to construct optimized theorem provers for all of them. The advantage of the second approach is that only one theorem prover needs to be written which can be reused for all translatable modal logics. In addition, a translation method can be expected to be more robust against small changes in the logic. Therefore translating seems to be the more sensible approach for most modal logics, with the exception of only a few main ones.

Translation of modal logics into first-order logic, with the explicit goal to mechanize such logics is an approach that has been introduced in (Morgan, 1976). Morgan distinguishes two types of translations: the *semantical translation*, which is nowadays known as the *relational translation* (see e.g., (Fine, 1975; van Benthem, 1976; Moore, 1977)) and the *syntactic* translation, which consists in reifing modal formulae (i.e., transforming them into first-order terms) and in translating the axioms and inference rules from a Hilbert-style system into classical logic using an additional provability predicate symbol. This is also sometimes called *reflection.* With such a syntactic translation, every propositional normal modal logic with a finite axiomatization can be translated into classical predicate logic. However, using this general translation, decidability of modal logics is lost in general, although the work in (Hustadt and Schmidt, 2003) has found a way to avoid this problem for many standard modal logics. We will study relational translations in this paper, which instead of simply translating a modal formula into full first-order logic, can translate modal formulas into a decidable subset of first-order logic. The fragment that we will be using is GF^2, the intersection of the 2-variable fragment (Grädel et al., 1997) and the guarded fragment (Andreka et al., 1998). We modify the relational translation in such a way that explicit translation of frame properties can be avoided. In this way, many modal logics with frame properties outside GF^2 can be translated into GF^2.

A survey on translation methods for modal logics can be found in (Ohlbach et al., 2001), where more references are provided, for instance about the functional translation (see e.g., (Herzig, 1989; Ohlbach, 1993; Nonnengart, 1996)), see also in (Orłowska, 1988; D'Agostino et al., 1995) for other types of translations.

Guarded fragments. Both the guarded fragment, introduced in (Andreka et al., 1998) (see also (de Nivelle, 1998; Grädel, 1999b; Ganzinger and de Nivelle, 1999; de Nivelle et al., 2000; de Nivelle and de Rijke, 2003)) and FO^2, the fragment of classical logic with two variables

(Gabbay, 1981; Grädel et al., 1997; de Nivelle and Pratt-Hartmann, 2001), have been used for the purpose of 'hosting' translations of modal formulas. The authors of (Andreka et al., 1998) explicitly mention the goal of identifying 'the modal fragment of first-order logic' as a motivation for introducing the guarded fragment. Apart from having nice logical properties (Andreka et al., 1998), the guarded fragment GF has an EXPTIME-complete satisfiability problem when the maximal arity of the predicate symbols is fixed in advance (Grädel, 1999b). Hence its worst-case complexity is identical to some simple extensions of modal logic K, as for example the modal logic K augmented with the universal modality (Spaan, 1993). Moreover, mechanization of the guarded fragment is possible thanks to the design of efficient resolution-based decision procedures (de Nivelle, 1998; Ganzinger and de Nivelle, 1999). In (Hladik, 2002), a tableau procedure for the guarded fragment with equality based on (Hirsch and Tobies, 2002) is implemented and tested; see also a prover for FO^2 described in (Marx et al., 1999).

However, there are some simple modal logics with the satisfiability problem in PSPACE ((Ladner, 1977)) that cannot be translated into GF through the relational translation. The reason for this is the fact that the frame condition that characterizes the logic cannot be expressed in GF. The simplest example of such a logic is probably S4 which is characterized by reflexivity and transitivity. Many other examples will be given throughout the paper. Adding transitivity axioms to a GF-formula causes undecidability (see (Grädel, 1999a)).

Because of the apparent insufficiency of GF to capture basic modal logics, various extensions of GF have been proposed and studied. In (Ganzinger et al., 1999), it was shown that GF^2 with transitivity axioms is decidable, on the condition that binary predicates occur only in guards. The complexity bound given there is non-elementary, which makes the fragment not very relevant to deal with logics, say in EXPTIME.

In (Szwast and Tendera, 2001), the complexity bound for GF^2 with transitive guards is improved to 2EXPTIME and it was shown 2EXPTIME-hard in (Kieronski, 2003). As a consequence, the resulting strategy is not the most efficient strategy to mechanize modal logics with transitive relations (such as S4)

Another fragment was explored in (Grädel and Walukiewicz, 1999), see also (Grädel, 1999a). There it was shown that μGF, the guarded fragment extended with a μ-calculus-style fixed point operator is still decidable and in 2EXPTIME. This fragment does contain the simple modal logic S4, but the machinery is much more heavy than a direct decision procedure would be. After all, there exist simple tableaux procedures for S4. In addition, μGF does not have the finite model property, although S4 has.

Almost structure-preserving translations. In this paper, we put emphasis on the fact that GF^2 is a sufficiently well-designed fragment of classical logic for dealing with a large variety of modal logics. An approach that seems better suited for theorem proving than the translation into the rich logic μGF, and that does more justice to the low complexities of simple modal logics is the approach taken in (de Nivelle, 1999; de Nivelle, 2001). There, an almost structure-preserving translation from the modal logics S4, S5 and K5 into GF^2 was given. The subformulae of a modal formula are translated in the standard way, except for subformulae that are $\Box$-formulae. The translation of $\Box$-formulae $\Box\psi$ depends on the frame condition and encodes the propagation of single-steps constraints (as done in (Massacci, 2000)) so that ψ holds true in successor states. In (de Nivelle, 1999; de Nivelle, 2001), the translations and their correctness proofs were ad hoc, and it was not clear upon which principles they are based. In this paper we show that the almost structure preserving translation relies on the fact that the frame conditions for K4, S4 and K5 are *regular* in some sense that will be made precise in Section 2.2. The simplicity of the almost structure-preserving translation leaves hope that GF^2 may be rich enough after all to naturally capture most of the basic modal logics.

We call the translation method almost structure-preserving because it preserves the structure of the formula almost completely. Only for subformulae of the form $[a]\phi$ does the translation differ from the usual relational translation. On these subformulae, the translation simulates an NDFA based on the frame condition of the modal logic. In our view this translation also provides an explanation why some modal logics like S4, have nice tableau procedures (see e.g. (Heuerding et al., 1996; Goré, 1999; Massacci, 2000; del Cerro and Gasquet, 2002; del Cerro and Gasquet, 2004; Horrocks and Sattler, 2004)): the tableau rule for subformulae of form $[a]\phi$ can be viewed as simulating an NDFA, in the same way as the almost structure-preserving translation.

In this paper, we show that the methods of (de Nivelle, 1999) can be extended to a very large class of modal logics. Some of the modal logics in this class have frame properties that can be expressed only by recursive conditions, like for example transitivity. By a *recursive condition* we mean a condition that needs to be iterated in order to reach a fixed point. The class of modal logics that we consider is the class of *regular grammar logics with converse.* The axioms of such modal logics are of form $[a_0]\mathrm{p} \Rightarrow [a_1]\dots[a_n]\mathrm{p}$ where each $[a_i]$ is either a forward or a backward modality. Another condition called *regularity* is required and will be formally defined in Section 2.2.

With our translation, we are able to translate numerous modal logics into $GF^2 = FO^2 \cap GF$, despite the fact that their frame conditions are not expressible in $FO^2 \cup GF$. These logics include the standard modal logics K4, S4, K5, K45, S5, some information logics (see e.g. (Vakarelov, 1987)), nominal tense logics (see e.g. (Areces et al., 2000)), description logics (see e.g. (Sattler, 1996; Horrocks and Sattler, 1999)), propositional intuitionistic logic (see e.g. (Chagrov and Zakharyaschev, 1997)) and bimodal logics for intuitionistic modal logics $\mathbf{IntK}_{\Box} + \Gamma$ as those considered in (Wolter and Zakharyashev, 1997). Hence the main contribution of the paper is the design of a very simple and generic translation from regular grammar logics with converse into the decidable fragment of classical logic GF^2. The translation is easy to implement and it mimics the behavior of some tableaux-based calculi for modal logics. As a consequence, we are able to show that the source logics that can be translated into GF^2 have a satisfiability problem in EXPTIME. This allows us to establish such an upper bound uniformly for a very large class of modal logics, for instance for intuitionistic modal logics (another approach is followed in (Alechina and Shkatov, 2003) leading to less sharp complexity upper bounds). We are considering here the satisfiability problem. However because of the very nature of the regular grammar logics with converse, our results apply also to the global satisfiability problem and to the logical consequence problem.

We do not claim that for most source logics the existence of a transformation into GF^2 of low complexity is surprising at all. In fact it is easy to see that from each simple modal logic for instance in PSPACE there must exist a polynomial transformation into GF^2, because PSPACE is a subclass of EXPTIME. The EXPTIME-completeness of fixed-arity GF implies that there exists a polynomial time transformation from every logic in PSPACE into fixed-arity GF. It can even be shown that there exists a logarithmic space transformation. However, the translation that establishes the reduction would normally make use of first principles on Turing machines. Trying to efficiently decide modal logics through such a transformation would amount to finding an optimal implementation in Turing machines, which is no easier than a direct implementation on a standard computer.

Our paper also answers a question stated in (Demri, 2001): Is there a decidable first-order fragment, into which the regular grammar logics can be translated in a natural way? The translation method that we give in this paper suggests that GF^2 is the answer. It is too early to state that the transformation from regular grammar logics with converse into GF^2 defined in this paper can be used to mechanize efficiently such source logics with a prover for GF^2, but we show evidence that GF^2 is a most valuable decidable first-order fragment to translate modal logics into, even when their frame conditions are not expressible in GF^2.

Structure of the paper. The paper starts by introducing multimodal languages with converses, semi-Thue systems, and some formal language theory. Using this, we can define regular grammar logics in Section 2.2. The same section also contains examples of regular grammar logics, which show that there are some natural modal logics covered by our framework.

Section 2.3 starts by repeating a standard result about derivability in Hilbert-style systems. After that, we prove a new result which characterizes when a grammar rule is a consequence of a set of grammar rules. This characterization will be used in Section 4 for determining which grammars define the same logic. In addition, we prove a closure theorem, which will be used in Section 3.2.

Section 3.1 presents the translation into GF^2. In Section 3.2 it is proven correct. In Section 4, we explore the borders of the translation method, and state some conjectures concerning which classes of logics can be translated. Section 5 contains the comparison of related works with ours. Section 6 concludes the paper and states some open questions, and future directions of research.

2. Multimodal Logic with Converse

We first introduce modal languages, after that we introduce modal frames and models. In standard modal logic, one has two operators $\Box\phi$, and $\Diamond\phi$, which denote that ϕ is true in all successor states, or true in at least one successor state. In multimodal logic, different types of successor relations are distinguished, which are labelled by elements of an alphabet Σ. As usual, an alphabet Σ is a finite set $\{a_1, \ldots, a_m\}$ of symbols. We write Σ^* to denote the set of finite strings that can be built from the elements of Σ, and we write ϵ for the empty string. We write $u_1 \cdot u_2$ for the concatenation of u_1 and u_2. For a string $u \in \Sigma^*$, we write $|u|$ to denote its length. A *language* over some alphabet Σ is defined as a subset of Σ^*.

Definition 2.1. We assume a countably infinite set PROP of propositional variables. Let Σ be an alphabet. The multimodal language ML$^\Sigma$ based on Σ is defined by the following schema:

$$\phi, \psi ::= \mathrm{p} \mid \bot \mid \top \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid [a]\phi \mid \langle a \rangle \phi$$

where p $\in$ PROP and $a \in \Sigma$ The $\Box$-formulae are the formulae of the form $[a]\psi$. We write $|\phi|$ to denote the *size* of the formula ϕ, that is the number of symbols needed to write ϕ down. A formula ϕ is in *negation normal form* (NNF) if $\neg$ occurs only in front of propositional variables. ∇

Without any loss of generality, we can make use of the NNF when we translate formulae to GF^2. The use is not essential, but it simplifies the presentation.

Definition 2.2. Let Σ be an alphabet. A Σ-*frame* is a pair $\mathcal{F} = \langle W, R \rangle$, such that W is a non-empty set, and R is a mapping from the elements of Σ to binary relations over W. We usually write R_a instead of $R(a)$. A Σ-*model* $\mathcal{M} = \langle W, R, V \rangle$ is obtained by adding a valuation function V with signature PROP $\rightarrow \mathcal{P}(W)$ to the frame. For every p $\in$ PROP, $V(\mathrm{p})$ denotes the set of worlds where p is true.
The satisfaction relation $\models$ is defined in the usual way:

- For every p $\in$ PROP, $\mathcal{M}, x \models \mathrm{p}$ iff $x \in V(\mathrm{p})$.
- For every $a \in \Sigma$, $\mathcal{M}, x \models [a]\phi$ iff for every y such that $R_a(x, y)$, $\mathcal{M}, y \models \phi$.
- For every $a \in \Sigma$, $\mathcal{M}, x \models \langle a \rangle\phi$ iff there is an y such that $R_a(x, y)$ and $\mathcal{M}, y \models \phi$.
- $\mathcal{M}, x \models \phi \wedge \psi$ iff $\mathcal{M}, x \models \phi$ and $\mathcal{M}, x \models \psi$.
- $\mathcal{M}, x \models \phi \vee \psi$ iff $\mathcal{M}, x \models \phi$ or $\mathcal{M}, x \models \psi$.
- $\mathcal{M}, x \models \neg\phi$ iff it is not the case that $\mathcal{M}, x \models \phi$.

A formula ϕ is said to be *true* in the Σ-model $\mathcal{M}$ (written $\mathcal{M} \models \phi$) iff for every $x \in W$, $\mathcal{M}, x \models \phi$. A formula ϕ is said to be *satisfiable* if there exist a Σ-model $\mathcal{M} = \langle W, R, V \rangle$ and $w \in W$, such that $\mathcal{M}, w \models \phi$. ∇

In order to be able to cope with properties such as symmetry and euclideanity, one needs to be able to express *converses*. Probably the most natural way to do this, is by extending the modal language with backward modal operators $[a]^{-1}\phi$ and $\langle a \rangle^{-1}\phi$. Unfortunately, this approach does not work for us, because we want to be able to express frame conditions using languages over Σ, and in this way the backward modalities have no counterpart in Σ.

Because of this, we follow another approach and we assume that to each a in the alphabet Σ, a unique converse symbol $\overline{a}$ is associated, which is also in Σ. In this way, one can partition Σ into two parts, the *forward* part and the *backward* part.

Definition 2.3. Let Σ be an alphabet. We call a function $\bar{\cdot}$ on Σ a *converse mapping* if for every $a \in \Sigma$, we have $\overline{a} \neq a$ and $\overline{\overline{a}} = a$. ∇

It is easy to prove the following result.

Lemma 2.1. Let Σ be an alphabet with converse mapping $\bar{\cdot}$. Then $\bar{\cdot}$ is a bijection on Σ. In addition, Σ can be partitioned into two disjoint sets Σ^+ and Σ^-, such that **(1)** for every $a \in \Sigma^+$, $\overline{a} \in \Sigma^-$, **(2)** for every $a \in \Sigma^-$, $\overline{a} \in \Sigma^+$.

In fact, there exist many partitions $\Sigma = \Sigma^- \cup \Sigma^+$. When we refer to such a partition, we assume that an arbitrary one is chosen. We call the modal operators indexed by letters in Σ^+ *forward modalities* (conditions on successor states) whereas the modal operators indexed by letters in Σ^- are called *backward modalities* (conditions on predecessor states).

Definition 2.4. Let Σ be an alphabet with converse mapping $\bar{\cdot}$. The converse mapping $\bar{\cdot}$ is extended to words over Σ^* as follows:

1. $\overline{\epsilon} \overset{\text{def}}{=} \epsilon$,
2. if $u \in \Sigma^*$ and $a \in \Sigma$, then $\overline{u \cdot a} \overset{\text{def}}{=} \overline{a} \cdot \overline{u}$.

∇

In order to ensure that converses behave like converses should, we impose the following, obvious constraint on the Σ-frames:

Definition 2.5. Let Σ be an alphabet with converse mapping $\bar{\cdot}$. We call a Σ-frame a $\langle \Sigma, \bar{\cdot} \rangle$-frame if, for every $a \in \Sigma$, $R_{\overline{a}}$ equals $\{\langle y, x \rangle \mid R_a(x,y)\}$. ∇

In the rest of the paper, we adopt the following working definition for a logic.

Definition 2.6. Let Σ be an alphabet with converse mapping $\bar{\cdot}$. A *logic* $\mathcal{L}$ is pair $\langle \mathrm{ML}^{\Sigma}, \mathcal{C} \rangle$ such that $\mathcal{C}$ is a class of $\langle \Sigma, \bar{\cdot} \rangle$-frames. A formula $\phi \in \mathrm{ML}^{\Sigma}$ is *$\mathcal{L}$-satisfiable* [resp. *$\mathcal{L}$-valid*] iff there exist a $\langle \Sigma, \bar{\cdot} \rangle$-model $\mathcal{M} = \langle W, R, V \rangle$ and $w \in W$ such that $\mathcal{M}, w \models \phi$ and $\langle W, R \rangle \in \mathcal{C}$ [resp. $\neg\phi$ is not $\mathcal{L}$-satisfiable]. ∇

2.1. Defining Logics by Production Rules

We want to study validity and satisfiability in various modal logics. Modal logics are traditionally defined by subclasses of frames (see Definition 2.6), or by axioms. For example, the logic S4 can be either defined by the modal axioms $[a]\phi \rightarrow [a][a]\phi$ and $[a]\phi \rightarrow \phi$ or by the subclass of frames in which all relations R_a are reflexive and transitive. For many modal logics, the axioms stand in natural correspondence to the condition that has to be imposed on the frames.

We will use a language theoretical framework for defining frames conditions. Since the accessibility relations are labelled by letters, paths through a frame can be labelled by words. Using this, certain conditions on the accessibility relation can be represented by production rules. For example, the transitivity rule $\forall xyz\ \mathbf{R}_a(x,y) \wedge \mathbf{R}_a(y,z) \Rightarrow \mathbf{R}_a(x,z)$ can be represented by the rule $a \rightarrow a \cdot a$. Similarly, the implication $\forall xyz\ \mathbf{R}_a(x,y) \wedge \mathbf{R}_b(y,z) \Rightarrow \mathbf{R}_c(x,z)$ can be represented by the rule $c \rightarrow a \cdot b$. In order to formally define how a frame satisfies a production rule, we need the following definition.

Definition 2.7. Let Σ be an alphabet and $\mathcal{F} = \langle W, R\rangle$ be a Σ-frame. The interpretations R_a are recursively extended to words $u \in \Sigma^*$ as follows:

- $R_\epsilon \overset{\text{def}}{=} \{\langle x,x\rangle \mid x \in W\}$,
- for all $u \in \Sigma^*$ and $a \in \Sigma$,

$$R_{u\cdot a} \overset{\text{def}}{=} \{\langle x,y\rangle \mid \exists z \in W,\ R_u(x,z) \text{ and } R_a(z,y)\}.$$

∇

Definition 2.8. Let Σ be an alphabet with converse mapping $\bar{\cdot}$. A *semi-Thue system* S over Σ is a set of *production rules* of form $u \rightarrow v$ with $u, v \in \Sigma^*$. ∇

A semi-Thue system is similar to a grammar, but it has no start symbol, and there is no distinction between terminal and non-terminal symbols. Using semi-Thue systems, we can define more precisely how semi-Thue systems encode conditions on Σ-frames.

Definition 2.9. Let $u \to v$ be a production rule over some alphabet Σ with converse mapping $\bar{\cdot}$. We say that a $\langle \Sigma, \bar{\cdot} \rangle$-frame $\mathcal{F} = \langle W, R \rangle$ *satisfies* $u \to v$ if the inclusion $R_v \subseteq R_u$ holds. A $\langle \Sigma, \bar{\cdot} \rangle$-frame $\mathcal{F}$ satisfies a *semi-Thue system* S if it satisfies each of its rules. We also say that the production rule (or the semi-Thue system) *is true* in $\mathcal{F}$. ∇

Observe that in Definition 2.9, u and v are swapped when passing from the production rule to the relation inclusion.

Definition 2.10. A formula ϕ is said to be S-*satisfiable* iff there is a $\langle \Sigma, \bar{\cdot} \rangle$-model $\mathcal{M} = \langle W, R, V \rangle$ which satisfies S, and which has an $x \in W$ such that $\mathcal{M}, x \models \phi$. Similarly, a formula ϕ is said to be S-*valid* iff in all $\langle \Sigma, \bar{\cdot} \rangle$-models $\mathcal{M} = \langle W, R, V \rangle$ that satisfy S, for every $x \in W$, we have $\mathcal{M}, x \models \phi$. ∇

Transitivity on the relation R_a can be expressed by the semi-Thue system $\{a \to a \cdot a\}$. Similarly, reflexivity can be expressed by the system $\{a \to \epsilon\}$.

Semi-Thue systems are obviously related to formal grammars, but in a semi-Thue system, the production rules are used for defining a relation between words, rather than for defining a subset of words. The former is precisely what we need to define grammar logics.

Definition 2.11. A multimodal logic $\langle \mathrm{ML}^{\Sigma}, \mathcal{C} \rangle$ with Σ an alphabet with converse mapping $\bar{\cdot}$ is said to be a *grammar logic with converse* if there is a finite semi-Thue system S over Σ such that $\mathcal{C}$ is the set of $\langle \Sigma, \bar{\cdot} \rangle$-frames satisfying S. ∇

We will mostly omit the suffix 'with converse', because we study only grammar logics with converse in this paper. One could give Definition 2.9 without converse, but this will bring no increased generality, because a logic without converse can always be viewed as a sublogic of a logic with converse. Consider a grammar logic $\mathcal{L}$ without converse defined by a semi-Thue system S over alphabet Σ. One can put $\Sigma' = \Sigma \cup \{\overline{a} \mid a \in \Sigma\}$, and for each $a \in \Sigma$, put $\overline{\overline{a}} = a$. Each Σ-frame can now be obviously extended to a $\langle \Sigma', \bar{\cdot} \rangle$-frame.

The modal logic S4 can be defined by the context-free semi-Thue system $\{a \to \epsilon,\ a \to aa\}$. The modal logic B can be defined by $\{a \to \overline{a}\}$. The following correspondence result is standard, see for example (van Benthem, 1984).

Theorem 2.2. Let Σ be an alphabet with converse mapping $\bar{\cdot}$, and S be a semi-Thue system over Σ. The following statements are equivalent:

1. In every $\langle \Sigma, \bar{\cdot} \rangle$-frame $\mathcal{F}$ satisfying S, for every p $\in$ PROP, $[u]\mathrm{p} \Rightarrow [v]\mathrm{p}$ is valid.
   For a word $u = (u_1, \ldots, u_m)$, $[u]\mathrm{p}$ is an abbreviation for $[u_1] \cdots [u_m]\mathrm{p}$.
2. $R_v \subseteq R_u$ in every $\langle \Sigma, \bar{\cdot} \rangle$-frame satisfying S.
   This is the same as saying that $\mathcal{F}$ makes $u \to v$ true.

Originally, grammar logics were defined with formal grammars in (del Cerro and Penttonen, 1988) (as in (Baldoni, 1998; Demri, 2001; Demri, 2002)), and they form a subclass of Sahlqvist modal logics (Sahlqvist, 1975) with frame conditions expressible in Π_1 when S is context-free (see e.g. Definition 2.13). Π_1 is the class of first-order formulae of the form $\forall\ x_1\ \forall\ x_2\ \ldots \forall\ x_n\ \phi$ where ϕ is quantifier-free. In the present paper, we adopt a lighter presentation based on semi-Thue systems as done in (Chagrov and Shehtman, 1994), which is more appropriate.

2.2. Regular grammar logics with converse

In order to define the class of regular grammar logics with converse, we need to recall a few notions from formal language theory.

Definition 2.12. Let S be a semi-Thue system. The one-step derivation relation $\Rightarrow_{\mathrm{S}}$ based on S is defined as follows: $u \Rightarrow_{\mathrm{S}} v$ iff there exist $u_1, u_2 \in \Sigma^*$, and $u' \to v' \in \mathrm{S}$, such that $u = u_1 \cdot u' \cdot u_2$, and $v = u_1 \cdot v' \cdot u_2$. The full derivation relation $\Rightarrow^*_{\mathrm{S}}$ is defined as the reflexive and transitive closure of $\Rightarrow_{\mathrm{S}}$. For every $u \in \Sigma^*$, we write $\mathrm{L}_{\mathrm{S}}(u)$ to denote the language $\{v \in \Sigma^* \mid u \Rightarrow^*_{\mathrm{S}} v\}$. ∇

Definition 2.13. The system S is *context-free* if all production rules are of the form $a \to v$ with $a \in \Sigma$ and $v \in \Sigma^*$. A context-free semi-Thue system S, based on Σ, is called *regular* if for every $a \in \Sigma$, the language $\mathrm{L}_{\mathrm{S}}(a)$ is regular. In that case, we assume there is a function that associates to each $a \in \Sigma$, an automaton $\mathcal{A}_a$ that accepts $\mathrm{L}_{\mathrm{S}}(a)$.

The *converse closure* $\overline{\mathrm{S}}$ of a system S over an alphabet Σ with converse mapping $\bar{\cdot}$ is the semi-Thue system $\{\overline{u} \to \overline{v} : u \to v \in \mathrm{S}\}$. A system S is said to be *closed under converse* if $\mathrm{S} = \overline{\mathrm{S}}$. ∇

Regular languages can be recognized by finite-state automata. We recall the definition of finite-state automaton, so that we can refer to it later.

Definition 2.14. A *non-deterministic finite automaton* (NDFA) $\mathcal{A}$ is defined by a tuple (Q, s, F, δ). Here Q is the finite, non-empty set of states. $s \in Q$ is the *initial* state. $F \subseteq Q$ is the set of *accepting* states. $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ is the *transition relation.*
The *extension* δ^* of δ is recursively defined as follows:

– For every state $q \in Q$, $\langle q, \epsilon, q \rangle \in \delta^*$.

– For all strings u and states q, q', q'', if $\langle q, u, q' \rangle \in \delta^*$ and $\langle q', \epsilon, q'' \rangle \in \delta$, then $\langle q, u, q'' \rangle \in \delta^*$.

– For all strings u, letters a, and states q, q', q'', if $\langle q, u, q' \rangle \in \delta^*$ and $\langle q', a, q'' \rangle \in \delta$, then $\langle q, u \cdot a, q'' \rangle \in \delta^*$.

$\mathcal{A}$ *accepts* a word u if there is a state $q \in F$, such that $\langle s, u, q \rangle \in \delta^*$. We write $\mathrm{L}(\mathcal{A})$ to denote the set of finite words accepted by $\mathcal{A}$. A language L is *regular* if there exists an NDFA $\mathcal{A}$, such that $\mathrm{L} = \{u \mid \mathcal{A} \text{ accepts } u\}$. ∇

For more details on NDFA's, we refer to (Hopcroft and Ullman, 1979). In Definition 2.13, we do not specify which automaton $\mathcal{A}_a$ is associated to a.

Definition 2.15. A multimodal logic $\langle \mathrm{ML}^{\Sigma}, \mathcal{C} \rangle$ with Σ an alphabet with converse mapping $\bar{\cdot}$ is said to be a *regular grammar logic with converse* if there is a finite regular semi-Thue system S over Σ closed under converse such that $\mathcal{C}$ is the set of $\langle \Sigma, \bar{\cdot} \rangle$-frames satisfying S. ∇

Example 2.1. The standard modal logics K, T, B, S4, K5, K45, and S5 can be defined as regular grammar logics over the singleton alphabet $\Sigma = \{a\}$. In Table I, we specify the semi-Thue systems through regular expressions for the languages $\mathrm{L}_{\mathrm{S}}(a)$.
$\lhd$

Numerous other logics for specific application domains are in fact regular grammar logics with converse, or logics that can be reduced to such logics. We list below some examples:

– description logics (with role hierarchy, transitive roles), see e.g. (Horrocks and Sattler, 1999).
– knowledge logics, see e.g. $\mathrm{S5}_m(\mathrm{DE})$ in (Fagin et al., 1995).
– bimodal logics for intuitionistic modal logics of the form $\mathbf{IntK}_{\Box} + \Gamma$ (Wolter and Zakharyashev, 1997). Indeed, let S be a regular semi-Thue system (over Σ) closed under converse and let $\Sigma' \subset \Sigma$ be such that for every $a \in \Sigma$, either $a \not\in \Sigma'$ or $\overline{a} \not\in \Sigma'$. Then, the semi-Thue system $\mathrm{S} \cup \{b \rightarrow bab, \overline{b} \rightarrow \overline{b}\overline{a}\overline{b} \mid a \in \Sigma'\}$ over $\Sigma \cup \{b, \overline{b}\}$ is also regular, assuming $b, \overline{b} \not\in \Sigma$. By taking advantage of (Ganzinger et al., 1999), in (Alechina and Shkatov, 2003) decidability of intuitionistic modal logics is also shown in a uniform manner.
– fragments of logics designed for the access control in distributed systems (Abadi et al., 1993; Massacci, 1997).

Table I. Regular languages for standard modal logics

logic	$\mathrm{L_S}(a)$	frame condition
K	$\{a\}$	(none)
KT	$\{a, \epsilon\}$	reflexivity
KB	$\{a, \overline{a}\}$	symmetry
KTB	$\{a, \overline{a}, \epsilon\}$	refl. and sym.
K4	$\{a\} \cdot \{a\}^*$	transitivity
KT4 = S4	$\{a\}^*$	refl. and trans.
KB4	$\{a, \overline{a}\} \cdot \{a, \overline{a}\}^*$	sym. and trans.
K5	$(\{\overline{a}\} \cdot \{a, \overline{a}\}^* \cdot \{a\}) \cup \{a\}$	euclideanity
KT5 = S5	$\{a, \overline{a}\}^*$	equivalence rel.
K45	$(\{\overline{a}\}^* \cdot \{a\})^*$	trans. and eucl.

- extensions with the universal modality (Goranko and Passy, 1992). Indeed, for every regular grammar logic with converse, its extension with a universal modal operator is also a regular grammar logic with converse by using simple arguments from (Goranko and Passy, 1992) (add a new letter U such that $[U]$ is an S5 modality and $[U]\mathrm{p} \Rightarrow [a]\mathrm{p}$ is a modal axiom for every letter a). Hence, satisfiability, global satisfiability and logical consequence can be handled uniformly with no increase of worst-case complexity.
- information logics, see e.g. (Vakarelov, 1987). For instance, the Nondeterministic Information Logic NIL introduced in (Vakarelov, 1987; Demri, 2000) can be shown to be a fragment of a regular grammar logic with converse with $\Sigma^+ = \{\mathrm{fin}, \mathrm{sim}\}$ and the production rules below (augmented with the converse closure):

  - $\mathrm{fin} \rightarrow \mathrm{fin} \cdot \mathrm{fin}$, $\mathrm{fin} \rightarrow \epsilon$,
  - $\mathrm{sim} \rightarrow \overline{\mathrm{sim}}$, $\mathrm{sim} \rightarrow \epsilon$,
  - $\mathrm{sim} \rightarrow \overline{\mathrm{fin}} \cdot \mathrm{sim} \cdot \mathrm{fin}$.

  For instance $\mathrm{L_S}(\mathrm{sim}) = \{\overline{\mathrm{fin}}\}^* \cdot \{\mathrm{sim}, \overline{\mathrm{sim}}, \epsilon\} \cdot \{\mathrm{fin}\}^*$.

Assuming that $\mathcal{L} = \langle \mathrm{ML}^{\Sigma}, \mathcal{C}_{\mathrm{S}} \rangle$ is a grammar logic with converse, checking whether $\mathcal{L}$ is regular is not an easy task. It is undecidable to check whether a context-free semi-Thue system is regular since it is undecidable whether the language generated by a linear grammar is regular (see e.g. (Mateescu and Salomaa, 1997, page 31)). However, if S is closed under converse and all the production rules in S are either right-linear or left-linear, then $\mathcal{L}$ is regular. We recall that S is right-linear if there is a partition $\{V, T\}$ of Σ such that the production rules in S are in $V \rightarrow T^* \cdot (V \cup \{\epsilon\})$. Similarly, S is left-linear if there is a partition $\{V, T\}$ of Σ such that the production rules are in $V \rightarrow (V \cup \{\epsilon\}) \cdot T^*$. Also, regularity is guaranteed if one can show that for every $a \in \Sigma$, the language $\mathrm{L_S}(a)$ is regular. All the modal logics cited above fall in this category. However, there is a remaining possible situation which is quite interesting. It might be the case that for some $a \in \Sigma$, the language $\mathrm{L_S}(a)$ is not regular but that there is another semi-Thue system S' s.t. $\langle \mathrm{ML}^\Sigma, \mathcal{C}_{\mathrm{S}'} \rangle$ defines the same logic as $\langle \mathrm{ML}^\Sigma, \mathcal{C}_{\mathrm{S}} \rangle$ and all $\mathrm{L_S}(a)$ are regular. This topic will be discussed in Section 4. In full generality, one should not expect to find a way to compute effectively a regular system S' but this shows the large scope of our translation.

2.3. Characterizations of Consequences

In this section we study the following two questions. Let Σ be an alphabet and S be a finite context-free semi-Thue system over Σ.

1. Which formulas are true in all $\langle \Sigma, \bar{\cdot} \rangle$-frames that satisfy S?
2. Which production rules $u \rightarrow v$ are true in all $\langle \Sigma, \bar{\cdot} \rangle$-frames that satisfy S?

The first question can be answered in the standard way by Hilbert-style deduction systems.

Definition 2.16. Let Σ be an alphabet with converse mapping $\bar{\cdot}$. Let S be semi-Thue system over Σ. The set of *derivable formulas* $\mathcal{H}$ is recursively defined as follows:

1. If ϕ is a propositional tautology, then $\phi \in \mathcal{H}$.
2. For all formulae ϕ and letters $a \in \Sigma$, $\quad [a]\phi \leftrightarrow \neg\langle a \rangle \neg\phi \in \mathcal{H}$.
3. If $\phi \in \mathcal{H}$, and $a \in \Sigma$, then also $[a]\phi \in \mathcal{H}$.
4. For all formulae ϕ, ψ and letters $a \in \Sigma$, $[a]\phi \wedge [a](\phi \Rightarrow \psi) \Rightarrow [a]\psi \in \mathcal{H}$,
5. For all formulas ϕ and letters $a \in \Sigma$, $\langle a \rangle [\overline{a}]\phi \rightarrow \phi \in \mathcal{H}$.
6. For every rule $u \rightarrow v \in \mathrm{S}$, for every formula ϕ,

$$[u]\phi \rightarrow [v]\phi \in \mathcal{H} \text{ and } [\overline{u}]\phi \rightarrow [\overline{v}]\phi \in \mathcal{H}.$$

∇

The following follows from (Sahlqvist, 1975).

Theorem 2.3. Let Σ be an alphabet with converse mapping $\bar{\cdot}$. Let S be a semi-Thue system over Σ. Then $\phi \in \mathcal{H}$ if and only if ϕ is true in all $\langle \Sigma, \bar{\cdot} \rangle$-frames satisfying S.

Regarding the second question, it is quite easy to see that $u \Rightarrow^*_{\mathrm{S}} v$ implies that $u \rightarrow v$ is true in every $\langle \Sigma, \bar{\cdot} \rangle$-frame that satisfies S. The converse does not always hold as shown in Example 2.2 below. When there is no converse mapping, it is indeed the case that all rules of form $u \rightarrow v$, which are true in all $\langle \Sigma, \bar{\cdot} \rangle$-frames, are derivable as $u \Rightarrow^*_{\mathrm{S}'} v$. This follows from (Chagrov and Shehtman, 1994, Theorem 3) (see also the tableaux-based proof in (Baldoni, 1998)) and it is related to the fact that every ordered monoid is embeddable into some ordered monoid of binary relations (see more details in (Chagrov and Shehtman, 1994)).

Example 2.2. Consider the semi-Thue system $\mathrm{S} = \{a \rightarrow \overline{a},\ b \rightarrow a^3\ \}$. In this system, $b \not\Rightarrow^*_{\mathrm{S}} a$. However, the rule $a \rightarrow \overline{a}$ expresses symmetry, which means that in a $\langle \Sigma, \bar{\cdot} \rangle$-frame satisfying S, whenever $\langle x, y \rangle \in R_a$, then also $\langle x, y \rangle \in R_{a^3}$. Therefore, $R_a \subseteq R_b$. One may think that the situation improves when the converse rules are added to S, but if one puts $\mathrm{S}' = \{a \rightarrow \overline{a},\ \ \overline{a} \rightarrow a,\ \ b \rightarrow a^3,\ \ \overline{b} \rightarrow \overline{a}^3\}$, then still $b \not\Rightarrow^*_{\mathrm{S}'} a$.

The production rule $b \Rightarrow^*_{S} a$ can be derived as follows: whenever $\langle x, y \rangle \in R_a$, then $\langle y, x \rangle \in R_{\overline{a}}$. As a consequence, $\langle x, y \rangle \in R_{a\overline{a}a}$. This means that the (non context-free) production rule $a\overline{a}a \rightarrow a$ is true in every frame. By combining $b \rightarrow aaa$, $\ \ a \rightarrow \overline{a}$, and $a\overline{a}a \rightarrow a$, we can derive $b \rightarrow a$. $\triangleleft$

In the sequel, we provide a complete characterization of the production rules that follow from a semi-Thue system S inspired from the (non context-free) rules added in Example 2.2.

Definition 2.17. Let Σ be an alphabet with converse mapping $\bar{\cdot}$. The *expansion system* E_Σ of Σ is the semi-Thue system

$$\{\ u \cdot \overline{u} \cdot u \rightarrow u \mid u \in \Sigma^* \backslash \{\epsilon\}\ \ \}.$$

∇

Lemma 2.4. Let Σ be an alphabet with converse mapping $\bar{\cdot}$.

(I) Every production rule in E_Σ is true in all $\langle \Sigma, \bar{\cdot} \rangle$-frames.

(II) For all $\langle\Sigma,\bar{\cdot}\rangle$-frames $\mathcal{F}$ and production rules $u \to v$, $\mathcal{F}$ satisfies $u \to v$ iff $\mathcal{F}$ satisfies $\overline{u} \to \overline{v}$.

(III) Let S be a semi-Thue system over Σ, u, v be two strings verifying $u \Rightarrow_{\mathrm{S}}^{*} v$, and $\mathcal{F}$ be a $\langle\Sigma,\bar{\cdot}\rangle$-frame satisfying S. Then $\mathcal{F}$ satisfies $u \to v$.

Proof. (I), (II) and (III) are by an easy verification. For instance, (III) can be proven by induction on the i for which $u \Rightarrow_{\mathrm{S}}^{i} v$. □

Theorem 2.5. Let Σ be an alphabet with converse mapping $\bar{\cdot}$ and S be a context-free semi-Thue system over Σ. Then, for all strings $u, v \in \Sigma^*$, the following are equivalent:

1. In every $\langle\Sigma,\bar{\cdot}\,\rangle$-frame $\mathcal{F} = \langle W, R\rangle$ that satisfies S, we have $R_v \subseteq R_u$.
2. There exists a string $z \in \Sigma^*$ such that $u \Rightarrow_{\mathrm{S}\cup\overline{\mathrm{S}}}^{*} z$ and $z \Rightarrow_{E_\Sigma}^{*} v$.
3. $u \Rightarrow_{\mathrm{S}\cup\overline{\mathrm{S}}\cup E_\Sigma}^{*} v$.

It is easy to see that **(2)** implies **(3)**. It follows from Lemma 2.4 that **(3)** implies **(1)**. We will use the rest of this section to show that **(1)** implies **(2)**. In order to do this, it is convenient to use the notion of *closure*. The closure will be also used in Section 3.2. If some $\langle\Sigma,\bar{\cdot}\rangle$-frame $\mathcal{F} = \langle W, R\rangle$ does not satisfy some context-free semi-Thue system, then one can add the missing edges to R and obtain a $\langle\Sigma,\bar{\cdot}\rangle$-frame that does satisfy the semi-Thue system. For context-free semi-Thue systems, one can define a function that assigns to each $\langle\Sigma,\bar{\cdot}\rangle$-frame the smallest frame that satisfies the semi-Thue system.

Definition 2.18. We first define an inclusion relation on $\langle\Sigma,\bar{\cdot}\rangle$-frames. Let Σ be an alphabet with converse mapping $\bar{\cdot}$. Let $\mathcal{F}_1 = \langle W, R_1\rangle$ and $\mathcal{F}_2 = \langle W, R_2\rangle$ be two $\langle\Sigma,\bar{\cdot}\rangle$-frames sharing the same set of worlds W. We say that $\mathcal{F}_1$ is a *subframe* of $\mathcal{F}_2$ if for every $a \in \Sigma$, $R_{1,a} \subseteq R_{2,a}$.

Using the inclusion relation, we define the *closure operator* C_{S} as follows: for a context-free semi-Thue system S over alphabet Σ with converse mapping $\bar{\cdot}$, for a $\langle\Sigma,\bar{\cdot}\rangle$-frame $\mathcal{F}$, the *closure* of $\mathcal{F}$ under S is defined as the smallest $\langle\Sigma,\bar{\cdot}\rangle$-frame that satisfies S, and which has $\mathcal{F}$ as a subframe. We write C_{S} for the closure operator. ∇

The closure always exists, and is unique, due to the Knaster-Tarski fixed point theorem. It can also be proven from Theorem 2.6, which states a crucial property of C_{S}, namely that every edge added by C_{S} can be justified by $\Rightarrow_{\mathrm{S}\cup\overline{\mathrm{S}}}^{*}$.

When S is regular, the map C_{S} is a monadic second-order definable graph transduction in the sense of (Courcelle, 1994) and it is precisely the inverse substitution h^{-1} in the sense of (Caucal, 2003) (see also (Caucal, 1996)) when the extended substitution h is defined by $a \in \Sigma \mapsto \mathrm{L}_{\mathrm{S}}(a)$.

Theorem 2.6. Let S be a context-free semi-Thue system over alphabet Σ with converse mapping $\bar{\cdot}$. Let $\mathcal{F} = \langle W, R\rangle$ be a $\langle \Sigma, \bar{\cdot}\rangle$-frame. For every letter $a \in \Sigma$, the relations R'_a of the $\langle \Sigma, \bar{\cdot}\rangle$-frame $\mathcal{F}' = C_{\mathrm{S}}(\mathcal{F}) = \langle W, R'\rangle$ are defined as follows:

$$R'_a = \{\langle x, y\rangle \mid \exists u \in \Sigma^* \text{such that } a \Rightarrow^*_{\mathsf{S}\cup\overline{\mathsf{S}}} u \text{ and } \langle x, y\rangle \in R_u\}.$$

Then $\mathcal{F}'$ is the closure of $\mathcal{F}$.

Proof. We have to show that

1. $\langle W, R'\rangle$ satisfies S,
2. $\langle W, R'\rangle$ is a $\langle \Sigma, \bar{\cdot}\rangle$-frame, and
3. among the $\langle \mathrm{S}, \bar{\cdot}\rangle$-frames that satisfy S and that have $\mathcal{F}$ as subframe, $\langle W, R'\rangle$ is the minimal such frame.

In order to show **(1)**, we show that for every rule $a \to u$ in S, the inclusion $R'_u \subseteq R'_a$ holds. Write $u = (u_1, \ldots, u_n)$ with $n \geq 0$, and each $u_i \in \Sigma$. Let $\langle x, y\rangle \in R'_u$. We need to show that $\langle x, y\rangle \in R'_a$.
By definition, there are $z_1, \ldots, z_{n-1} \in W$, such that

$$\langle x, z_1\rangle \in R'_{u_1}, \quad \langle z_1, z_2\rangle \in R'_{u_2}, \ldots, \quad \langle z_{n-1}, y\rangle \in R'_{u_n}.$$

By construction of R', there are words $v_1, \ldots, v_n \in \Sigma^*$, such that $u_1 \Rightarrow^*_{\mathsf{S}\cup\overline{\mathsf{S}}} v_1, \quad \ldots, \quad u_n \Rightarrow^*_{\mathsf{S}\cup\overline{\mathsf{S}}} v_n$, and

$$\langle x, z_1\rangle \in R_{v_1}, \quad \langle z_1, z_2\rangle \in R_{v_2}, \ldots, \quad \langle z_{n-1}, y\rangle \in R_{v_n}.$$

As a consequence, $\langle x, y\rangle \in R_{v_1 \cdot \ldots \cdot v_n}$. Because $a \Rightarrow_{\mathsf{S}\cup\overline{\mathsf{S}}} u, \ u = (u_1, \ldots, u_n)$, and each $u_i \Rightarrow^*_{\mathsf{S}\cup\overline{\mathsf{S}}} v_i$, we also have $a \Rightarrow^*_{\mathsf{S}\cup\overline{\mathsf{S}}} v_1 \cdot \ldots \cdot v_n$. It follows that $\langle x, y\rangle \in R'_a$, from the way R'_a was constructed.
Next we show **(2)**. As a preparation, it can be shown by induction that $a \Rightarrow^*_{\mathsf{S}\cup\overline{\mathsf{S}}} u$ iff $\overline{a} \Rightarrow^*_{\mathsf{S}\cup\overline{\mathsf{S}}} \overline{u}$. We need to show that for every $a \in \Sigma$,

$$\langle x, y\rangle \in R'_a \text{ iff } \langle y, x\rangle \in R'_{\overline{a}}.$$

$$\langle x, y\rangle \in R'_a \text{ iff}$$

$$\text{there exists a word } u \in \Sigma^*, \text{ for which } a \Rightarrow^*_{\mathsf{S}\cup\overline{\mathsf{S}}} u, \text{ and } \langle x, y\rangle \in R_u \text{ iff}$$

$$\text{there exists a word } \overline{u} \in \Sigma^*, \text{ for which } \overline{a} \Rightarrow^*_{\mathsf{S}\cup\overline{\mathsf{S}}} \overline{u} \text{ and } \langle y, x\rangle \in R_{\overline{u}} \text{ iff}$$

$$\langle y, x\rangle \in R_{\overline{a}}.$$

Finally we show **(3)**. Let $\langle W, R'' \rangle$ be a $\langle \Sigma, \bar{\cdot} \rangle$-frame, such that $\langle W, R \rangle$ is a subframe of $\langle W, R'' \rangle$ and $\langle W, R'' \rangle$ satisfies S. We want to show that for every $a \in \Sigma, \quad R'_a \subseteq R''_a$.

Assume that $\langle x, y \rangle \in R'_a$. This means that there exists an $u \in \Sigma^*$, for which $\langle x, y \rangle \in R_u$ and $a \Rightarrow^*_{\mathrm{S} \cup \overline{\mathrm{S}}} u$. From Lemma 2.4(II,III), we know that $a \rightarrow u$ is true in $\langle W, R \rangle$. Therefore, we have $\langle x, y \rangle \in R_a$. Because $R_a \subseteq R''_a$, we also have $\langle x, y \rangle \in R''_a$.
□
Actually, only Part 1 and Part 2 of Theorem 2.6 are needed in the proof of Theorem 2.5 and in Section 3.2.

We can now give the proof of Theorem 2.5 "(1) implies (2)". Let S be a context-free semi-Thue system. Let u, v be two words, such that in every $\langle \Sigma, \bar{\cdot} \rangle$-frame $\mathcal{F} = \langle W, R \rangle$ satisfying S, we have $R_v \subseteq R_u$.

Write $v = (v_1, \ldots, v_n)$ with $n \geq 0$. Let the frame $\mathcal{F}_v = \langle W, R \rangle$ be defined as follows:

- $W = \{w_1, \ldots, w_n, w_{n+1}\}$,
- If (and only if) $v_i = a$, then $\langle w_i, w_{i+1} \rangle \in R_a$, for $1 \leq i \leq n$ and $a \in \Sigma$.

Intuitively, the frame $\mathcal{F}_v$ consists of a single path, which is labelled with the word v. Let $\mathcal{F}'_v = \langle W, R' \rangle$ be obtained from $\mathcal{F}_v$ by the construction of Theorem 2.6, i.e. $\mathcal{F}'_v = C_{\mathrm{S}}(\mathcal{F}_v)$. Since by Theorem 2.6, $\mathcal{F}'_v$ is a frame satisfying S and by hypothesis $R'_v \subseteq R'_u$, we have $\langle w_1, w_{n+1} \rangle \in R'_u$.
If one writes $u = (u_1, \ldots, u_m)$, then there must exist $w'_1, \ldots, w'_{m+1} \in W$, such that

$$\langle w'_1, w'_2 \rangle \in R'_{u_1}, \quad \langle w'_2, w'_3 \rangle \in R'_{u_2}, \ldots, \langle w'_m, w'_{m+1} \rangle \in R'_{u_m},$$

with $w'_1 = w_1$ and $w'_{m+1} = w_{n+1}$.
By construction of $\mathcal{F}'_v$, there exist words $z_1, \ldots, z_m \in \Sigma^*$, such that

$$\langle w'_1, w'_2 \rangle \in R_{z_1}, \quad \langle w'_2, w'_3 \rangle \in R_{z_2}, \ldots, \langle w'_m, w'_{m+1} \rangle \in R_{z_m},$$

and

$$u_1 \Rightarrow^*_{\mathrm{S} \cup \overline{\mathrm{S}}} z_1, \quad u_2 \Rightarrow^*_{\mathrm{S} \cup \overline{\mathrm{S}}} z_2, \ldots, u_m \Rightarrow^*_{\mathrm{S} \cup \overline{\mathrm{S}}} z_m.$$

Therefore, for the word $z = z_1 \cdot z_2 \cdot \ldots \cdot z_m$, it follows that $\langle w_1, w_{n+1} \rangle \in R_z$ and $u \Rightarrow^*_{\mathrm{S} \cup \overline{\mathrm{S}}} z$. We will show that also $z \Rightarrow^*_{E_\Sigma} v$, from which then follows that $u \Rightarrow^*_{\mathrm{S} \cup \overline{\mathrm{S}} \cup E_\Sigma} v$.

Lemma 2.7. Let $\mathcal{F}_v = \langle W, R \rangle$ be the frame defined above from v. Let $z \in \Sigma^*$ be a string such that $\langle w_1, w_{n+1} \rangle \in R_z$. Then $z \Rightarrow^*_{E_\Sigma} v$.

The word z corresponds to a walk from w_1 to w_{n+1} in the frame $\mathcal{F}_v$. The frame $\mathcal{F}_v$ consists of a single path, which is labelled by the word v.

The word z is obtained by a walk on this path, which possibly changes direction a few times.

We call a maximal subpath that does not change direction *a segment*. A segment is either forward directed, or backward directed. So a segment s is of the form either

$$s = (w_i, w_{i+1}, \dots, w_{j-1}, w_j)$$

or

$$s = (w_i, w_{i-1}, \dots, w_{j+1}, w_j)$$

with $1 \leq i, j \leq n+1$ and $i \neq j$.
Using segments, the path can be written in the form

$$s_1 = (x_1, \dots, y_1), \dots, s_k = (x_k, \dots, y_k), \tag{1}$$

where

- all the states in the segments are in $\{w_1, \dots, w_{n+1}\}$,
- k is odd,
- for every i, $x_i \neq y_i$ and $x_{i+1} = y_i$ (assuming $i+1 \leq k$).

Given two states $w, w' \in \{w_1, \dots, w_{n+1}\}$, we write $w < w'$ whenever there are $1 \leq j < j' \leq n+1$ such that $w_j = w$ and $w_{j'} = w'$. Observe that if i is odd, then $x_i < y_i$. If i is even, then $x_i > y_i$. For every i, there is a unique string $v_i \in ((\Sigma^+)^* \cup (\Sigma^-)^*) \setminus \{\epsilon\}$ such that $\langle x_i, y_i \rangle \in R_{v_i}$. If i is odd, then v_i is a substring of v. If i is even, then v_i is a substring of $\overline{v}$. We call v_i *the associated string* of the segment $s_i = (x_i, \dots, y_i)$. We have $z = v_1 \cdot \ldots \cdot v_k$.

If $k = 1$, then $z = v_1 = v$, and we are done because $z \Rightarrow^0_{E_\Sigma} v$. Otherwise, let i with $1 < i < k$ be chosen in such a way that $(x_i, \dots, y_i)$ is a segment with minimal length. Such an i must exist, because $k \geq 3$. The segment $(x_{i-1}, \dots, y_{i-1})$ before $(x_i, \dots, y_i)$ cannot be strictly shorter than $(x_i, \dots, y_i)$. Suppose that it were. Then, if $i > 2$, the position $i-1$ would have been chosen instead of i. If $i = 2$, then $(x_2, \dots, y_2)$ is a segment that walks backwards in the direction of w_1. The first segment $(x_1, \dots, y_1)$ starts with $x_1 = w_1$ and must be at least as long, because otherwise $(x_2, \dots, y_2)$ would walk back through w_1, which is not possible because $\mathcal{F}_v$ is a chain starting in w_1.

For the same reason, the segment $(x_{i+1}, \dots, y_{i+1})$ cannot be strictly shorter than $(x_i, \dots, y_i)$. As a consequence, the neighbours of the i-th segment s_i are of the form

$$s_{i-1} = (x_{i-1}, \dots, y_{i-1}) = (x_{i-1}, \dots, y_i, \dots, x_i),$$

and

$$s_{i+1} = (x_{i+1}, \dots, y_{i+1}) = (y_i, \dots, x_i, \dots, y_{i+1}).$$

Then the associated strings are of the form

$$v_{i-1} = \alpha \cdot \overline{v_i} \ \text{ and } \ v_{i+1} = \overline{v_i} \cdot \beta, \ \text{for some } \alpha, \beta \in \Sigma^*.$$

The complete walk can be written as

$$s_1, \dots, (x_{i-1}, \dots, y_i, \dots, x_i), (x_i, \dots, y_i), (y_i, \dots, x_i, \dots, y_{i+1}), \dots s_k. \tag{2}$$

The complete string z is of the form

$$z = v_1 \cdot \dots \cdot \alpha \cdot \overline{v_i} \cdot v_i \cdot \overline{v_i} \cdot \beta \cdot \dots \cdot v_k,$$

which can be rewritten by the following rule, which is in E_Σ :

$$\overline{v_i} \cdot v_i \cdot \overline{v_i} \rightarrow \overline{v_i}.$$

The result is the string

$$v_1 \cdot \dots \cdot \alpha \cdot \overline{v_i} \cdot \beta \cdot \dots \cdot v_k. \tag{3}$$

If one replaces walk 2 by

$$s_1, \dots, (x_{i-1}, \dots, y_i, \dots, x_i, \dots, y_{i+1}), \dots, s_k,$$

then the result is a walk from w_1 to w_{n+1} with associated string 3. Since the walk consists of $k-2$ segments, it follows by induction that

$$v_1 \cdot \dots \cdot \alpha \cdot \overline{v_i} \cdot \beta \cdot \dots \cdot v_k \Rightarrow^*_{E_\Sigma} v,$$

from which follows that

$$v_1 \cdot \dots \cdot \alpha \cdot \overline{v_i} \cdot v_i \cdot \overline{v_i} \cdot \beta \cdot \dots \cdot v_k \Rightarrow^*_{E_\Sigma} v.$$

Example 2.3. Assume that $\Sigma = \{a, \overline{a}\}$, and that $\mathrm{S} = \{a \rightarrow \overline{a}\overline{a}a\}$. In every $\langle \Sigma, \bar{\cdot} \rangle$-frame in which S is true, also the production rule $a \rightarrow \overline{a}aa$ is true. This can be seen as follows. Let $\mathcal{F} = \langle W, R \rangle$ be a $\langle \Sigma, \bar{\cdot} \rangle$-frame for which $\langle x, y \rangle \in R_{\overline{a}aa}$. Then there are $w_1, w_2 \in W$, such that

$$\langle x, w_1 \rangle \in R_{\overline{a}}, \quad \langle w_1, w_2 \rangle \in R_a, \quad \langle w_2, y \rangle \in R_a.$$

Since $\mathcal{F}$ is a $\langle \Sigma, \bar{\cdot} \rangle$-frame, we have

$$\langle y, w_2 \rangle \in R_{\overline{a}}, \quad \langle w_2, w_1 \rangle \in R_{\overline{a}}, \quad \langle w_1, x \rangle \in R_a.$$

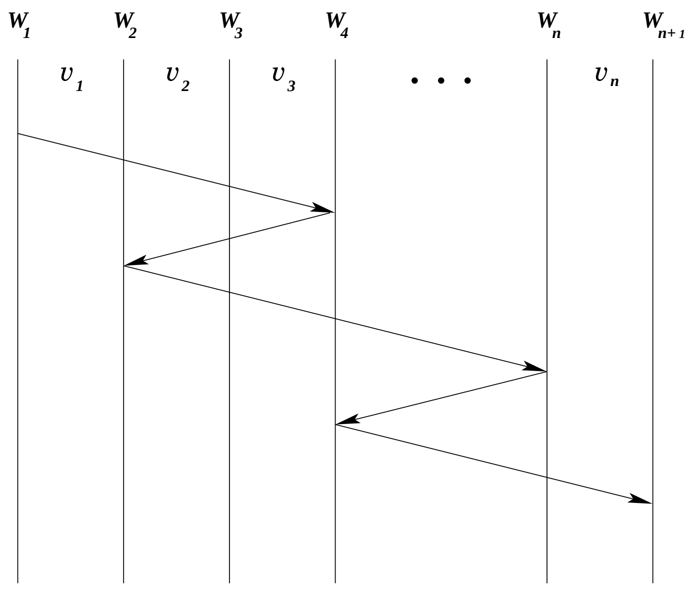


Figure 1. A walk from w_1 to w_{n+1} with 5 segments. In the corresponding string, it is possible to replace the substring $(v_2, v_3, \overline{v}_3, \overline{v}_2, v_2, v_3)$ by (v_2, v_3).

Because S is true in $\mathcal{F}$, also

$$\langle y, x\rangle \in R_a.$$

Now we have

$$\langle x, y\rangle \in R_{\overline{a}}, \quad \langle y, w_2\rangle \in R_{\overline{a}}, \quad \langle w_2, y\rangle \in R_a,$$

which implies $\langle x, y\rangle \in R_a$. Clearly not $a \Rightarrow_{\mathrm{S}}^{*} \overline{a}aa$, and also not $a \Rightarrow_{\mathrm{S}\cup\overline{\mathrm{S}}}^{*}$ $\overline{a}aa$. However, $a \Rightarrow_{\mathrm{S}} (\overline{a}\overline{a}a) \Rightarrow_{\overline{\mathrm{S}}} (\overline{a}aa)\overline{a}a \Rightarrow_{E_\Sigma} \overline{a}aa$. ◁

3. The Translation into GF^2

In this section, we define the transformation from regular grammar logics with converse into GF^2. The transformation can be carried out in logarithmic space. It behaves the same as the standard relational translation on all subformulae, with the exception of □-subformulae. On a □-subformula, it simulates the behaviour of an NDFA in order to determine to which worlds the $\Box$-formula applies. The translation generalises the results in (de Nivelle, 1999; de Nivelle, 2001) for the logics S4 and K5, which were at an ad hoc basis.

Unless otherwise stated, in the rest of this section, we assume that $\mathcal{L} = \langle \mathrm{ML}^{\Sigma}, \mathcal{C} \rangle$ is a regular grammar logic with converse such that $\mathcal{C}$ is the class of $\langle \Sigma, \bar{\cdot} \rangle$-frames satisfying S, a finite regular semi-Thue system closed under converse. For every $a \in \Sigma$, the automaton $\mathcal{A}_a$ is an NDFA recognizing the language $\mathrm{L}_{\mathrm{S}}(a)$. It would be possible to make $\mathcal{A}_a$ canonic, for example by defining $\mathcal{A}_a$ to be the minimal DFA accepting $\mathrm{L}_{\mathrm{S}}(a)$ -which is unique up to isomorphism- but there is no advantage in this. In contrast, as we shall see, this could even blow up the translation because an NDFA can have exponentially less states than a DFA accepting the same language (see e.g. (Hopcroft and Ullman, 1979)). It is in principle possible that $\mathcal{A}_a$ and $\mathcal{A}_{\overline{a}}$ are different automata, although they have to accept isomorphic languages (because $u \in \mathrm{L}_{\mathrm{S}}(a)$ iff $\overline{u} \in \mathrm{L}_{\mathrm{S}}(\overline{a})$ for every $u \in \Sigma^*$). We write $\mathcal{A}_a = (Q_a, s_a, F_a, \delta_a)$. When all rules in S are either right-linear or left-linear, then each automaton $\mathcal{A}_a$ can be effectively built in logarithmic space in $|\mathrm{S}|$, the size of S with some reasonably succinct encoding.

3.1. The Transformation

In the sequel we assume that the two variables in GF^2 are $\{x_0, x_1\}$. The symbols α and β are used as distinct meta-variables in $\{x_0, x_1\}$. Observe that in Definition 3.2 the quantification alternates over α and β.

In the translation, we use atomic formulae of the form $\mathbf{R}_a(\alpha, \beta)$ for every $a \in \Sigma$. Because of the conditions between R_a and $R_{\overline{a}}$ in $\langle \Sigma, \bar{\cdot} \rangle$-frames, we add the axioms of the form $\forall \alpha\beta \ \mathbf{R}_a(\alpha, \beta) \Leftrightarrow \mathbf{R}_{\overline{a}}(\beta, \alpha)$ which belong to GF^2. Although this treatment of converse relations allows us to avoid some case distinctions in the proofs, in practice we might adopt an alternative treatment with a smaller signature. Indeed, one can replace syntactically in the translation process $\mathbf{R}_{\overline{a}}(\alpha, \beta)$ by $\mathbf{R}_a(\beta, \alpha)$ for every $a \in \Sigma^+$.

Definition 3.1. We assume that to each letter $a \in \Sigma$, a unique binary predicate symbol $\mathbf{R}_a$ is associated. The formula CONV_{Σ} defined below deals with converses:

$$\mathrm{CONV}_{\Sigma} \overset{\mathrm{def}}{=} \bigwedge_{a \in \Sigma^+} \forall x_0 \ x_1 \ \mathbf{R}_a(x_0, x_1) \Leftrightarrow \mathbf{R}_{\overline{a}}(x_1, x_0).$$

∇

The formula CONV_{Σ} is in GF^2. When a subformula $[a]\phi$ is translated, it is replaced by formulae stating

At every point that is reachable through via a sequence of transitions labelled by a word in $\mathrm{L_S}(a)$ (i.e. accepted by the automaton $\mathcal{A}_a$), the translation of ϕ holds.

We define a function that takes two parameters, a one-place first-order formula and an NDFA. The result of the translation is a first-order formula (one-place again) that has the following meaning:

In every point that is reachable by a sequence of transitions labelled by a word that accepted by the automaton, the original one-place formula holds.

Definition 3.2. Let $\mathcal{A} = \langle Q, s, F, \delta \rangle$ be an NDFA and $\varphi(\alpha)$ be a *first-order* formula with one free variable α. Assume that for every state $q \in Q$, a fresh unary predicate symbol $\mathbf{q}$ is given. We define $t_{\mathcal{A}}(\alpha, \varphi)$ as the conjunction of the following formulas (the purpose of the first argument is to remember that α is the free variable of φ).

- For the initial state s, the formula $\mathbf{s}(\alpha)$ is included in the conjunction.
- For every transition $\langle q, a, r \rangle \in \delta$, the formula

$$\forall \alpha\beta \; [\; \mathbf{R}_a(\alpha, \beta) \Rightarrow (\mathbf{q}(\alpha) \Rightarrow \mathbf{r}(\beta)) \;]$$

  is included in the conjunction.
- For every ϵ-transition, $\langle q, \epsilon, r \rangle \in \delta$, the formula

$$\forall \alpha \; [\; (\mathbf{q}(\alpha) \Rightarrow \mathbf{r}(\alpha)) \;]$$

  is included in the conjunction.
- For each accepting state $q \in F$, the formula

$$\forall \alpha \; [\; \mathbf{q}(\alpha) \rightarrow \varphi(\alpha) \;]$$

  is included in the conjunction.

∇

The function $t_{\mathcal{A}}(\alpha, \psi)$ is applied on formulas ψ that are subformulae of an initial formula ϕ. Definition 3.2 requires that in each application of $t_{\mathcal{A}}$, distinct predicate symbols of form $\mathbf{q}$ for $q \in Q$ are introduced. This can be done either occurrence-wise, or subformula-wise. Occurrence-wise means that, if some subformula ψ of ϕ occurs more than once, then different fresh predicate symbols have to be introduced for each occurrence. Subformula-wise means that the different occurrences can share the fresh predicates. In the sequel, we adopt the subformula-wise approach. For every state $q \in Q$, we should write $\mathbf{q}_\varphi$ instead of $\mathbf{q}$ in the translation of $t_{\mathcal{A}}(\alpha, \varphi)$. We sometimes omit the subscript when it is not confusing.

If the automaton $\mathcal{A}$ has more than one accepting state, then $\varphi(\alpha)$ occurs more than once in the translation $t_{\mathcal{A}}(\alpha, \varphi)$. This may cause an exponential blow-up in the translation process but this problem can be easily solved by adding a new accepting state to the automaton, and adding ϵ-translations from the old accepting states into the new accepting state.

Now we can give the translation itself. It behaves like a standard relational translation on all subformulae, except for those of the form $[a]\psi$, on which $t_{\mathcal{A}_a}$ will be used. In order to easily recognize the $\Box$-subformulae, we require the formula ϕ to be in negation normal form. One could define the translation without it, but it would have more cases.

Definition 3.3. Let $\phi \in \mathrm{ML}^\Sigma$ be a modal formula in NNF. We define the translation $T_{\mathrm{S}}(\phi)$ as $t(\phi, x_0, x_1)$ from the following function $t(\psi, \alpha, \beta)$, which is defined by recursion on the subformulae ψ of ϕ :

- $t(\mathrm{p}, \alpha, \beta)$ equals $\mathbf{p}(\alpha)$, where $\mathbf{p}$ is a unary predicate symbol uniquely associated to the propositional variable p.
- $t(\neg \mathrm{p}, \alpha, \beta)$ equals $\neg \mathbf{p}(\alpha)$,
- $t(\psi \wedge \psi', \alpha, \beta)$ equals $t(\psi, \alpha, \beta) \wedge t(\psi', \alpha, \beta)$,
- $t(\psi \vee \psi', \alpha, \beta)$ equals $t(\psi, \alpha, \beta) \vee t(\psi', \alpha, \beta)$,
- for every $a \in \Sigma$, $t(\langle a \rangle \psi, \ \alpha, \beta)$ equals $\exists \beta \ [\ \mathbf{R}_a(\alpha, \beta) \wedge t(\psi, \beta, \alpha)\ ]$,
- for every $a \in \Sigma$, $t([a]\psi, \ \alpha, \beta)$ equals $t_{\mathcal{A}_a}(\alpha, t(\psi, \alpha, \beta)\ )$.

∇

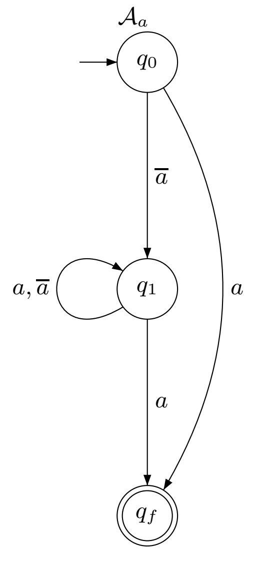
Hence, the first-order vocabulary used in $T_{\mathrm{S}}(\phi)$ includes

- unary predicate symbols $\mathbf{p}$ for every propositional variable p occurring in ϕ,
- binary predicate symbols $\mathbf{R}_a$ for every letter $a \in \Sigma$,
- unary predicate symbols $\mathbf{q}_{[a]\psi}$ for every $\Box$-subformula $[a]\psi$ of ϕ and $q \in Q_a$.

Lemma 3.1. Let $\phi \in \mathrm{ML}^\Sigma$ be a modal formula in NNF such that $m = \max\{|\mathcal{A}_a| \mid a \in \Sigma\}$.

1. The only variables occurring in $T_{\mathrm{S}}(\phi)$ are in $\{x_0, x_1\}$ and α is the only free variable in $t(\psi, \alpha, \beta)$.

Table II. The K5 automaton and $t_{\mathcal{A}_a}(\alpha, \varphi(\alpha))$ for arbitrary $\varphi(\alpha)$



$t_{\mathcal{A}_a}(\alpha, \varphi(\alpha))$

$\mathbf{q}_{0,\varphi}(\alpha)$

$\forall\alpha\beta\ [\ \mathbf{R}_a(\alpha,\beta) \Rightarrow (\mathbf{q}_{0,\varphi}(\alpha) \Rightarrow \mathbf{q}_{f,\varphi}(\beta))\ ]$

$\forall\alpha\beta\ [\ \mathbf{R}_{\overline{a}}(\alpha,\beta) \Rightarrow (\mathbf{q}_{0,\varphi}(\alpha) \Rightarrow \mathbf{q}_{1,\varphi}(\beta))\ ]$

$\forall\alpha\beta\ [\ \mathbf{R}_a(\alpha,\beta) \Rightarrow (\mathbf{q}_{1,\varphi}(\alpha) \Rightarrow \mathbf{q}_{1,\varphi}(\beta))\ ]$

$\forall\alpha\beta\ [\ \mathbf{R}_{\overline{a}}(\alpha,\beta) \Rightarrow (\mathbf{q}_{1,\varphi}(\alpha) \Rightarrow \mathbf{q}_{1,\varphi}(\beta))\ ]$

$\forall\alpha\beta\ [\ \mathbf{R}_a(\alpha,\beta) \Rightarrow \mathbf{q}_{1,\varphi}(\alpha) \Rightarrow \mathbf{q}_{f,\varphi}(\beta)\ ]$

$\forall\alpha\ [\ \mathbf{q}_{f,\varphi}(\alpha) \Rightarrow \varphi(\alpha)\ ]$

2. $T_{\mathrm{S}}(\phi)$ is in the guarded fragment.
3. The size of $T_{\mathrm{S}}(\phi)$ is in $\mathcal{O}(|\phi| \times m)$.
4. $T_{\mathrm{S}}(\phi)$ can be computed in logarithmic space in $|\phi| + m$.

When S is formed from production rules of a semi-Thue system that is either right-linear or left-linear, then m is in $\mathcal{O}(|\mathrm{S}|)$. For a given semi-Thue system S, the number m is fixed. As a consequence, $T_{\mathrm{S}}(\phi)$ has size linear in $|\phi|$ for a given logic. Observe also that $|\mathrm{CONV}_{\Sigma}|$ is a constant of the logic.

Example 3.1. Let $\phi = \Diamond\mathrm{p} \wedge \Diamond\Box\neg\mathrm{p}$ be the negation normal form of the formula $\neg(\Diamond\mathrm{p} \Rightarrow \Box\Diamond\mathrm{p})$. We consider K5, and assume one modality a, so $\Box$ is an abbreviation for $[a]$, and $\Diamond$ is an abbreviation for $\langle a \rangle$. Table II contains to the left an automaton $\mathcal{A}_a$ recognizing the language defined in Table I for K5 (page 78). To the right is the translation $t_{\mathcal{A}_a}(\alpha, \varphi(\alpha))$ for some first-order formula $\varphi(\alpha)$. The translation $T_{\mathrm{S}}(\phi)$ of ϕ is equal to

$$\exists\beta\ [\ \mathbf{R}_a(\alpha,\beta) \wedge \mathbf{p}(\beta)\ ] \wedge \exists\beta\ [\ \mathbf{R}_a(\alpha,\beta) \wedge t_{\mathcal{A}_a}(\beta, \mathbf{p}(\beta)\ )\ ].$$

$\lhd$

Since we perform the introduction of new symbols subformula-wise, it is possible to put the translation of the automaton outside of the translation of the modal formula. At the position where $t_{\mathcal{A}_a}(\alpha, t(\psi, \alpha, \beta))$ is translated, only $\mathbf{q_{0,\psi}}(\alpha)$ needs to be inserted where q_0 is the initial state of $\mathcal{A}_a$. The rest of the (translation of the) automaton yields an independent conjunct of translation.

Extension with nominals. The map T_{S} can be obviously extended to admit nominals in the language of the regular grammar logics with converse. The treatment of nominals can be done in the usual way by extending the definition of t as follows: $t(\mathbf{i}, \alpha, \beta) \overset{\text{def}}{=} \mathbf{c_i} = \alpha$ where $\mathbf{c_i}$ is a constant associated with the nominal $\mathbf{i}$. The target first-order fragment is GF^2 with constants and identity. For instance, nominal tense logics with transitive frames (see e.g., (Areces et al., 2000)), and description logics with transitive roles and converse (see e.g., (Sattler, 1996)), can be translated into $\mathrm{GF}^2[=]$ with constants in such a way. Additionnally, by using (Blackburn and Marx, 2002, Sect. 4) regular grammar logics with converse augmented with Gregory's "actually" operator (Gregory, 2001) can be translated into such nominal tense logics.

3.2. Satisfiability Preservation

We show that the map T_{S} preserves satisfiability. First, we introduce some notation. A *first-order model* is denoted by $\langle W, V \rangle$ where W is a non-empty set and V maps unary [resp. binary] predicate symbols into subsets of W [resp. $W \times W$]. Given a variable valuation $v : \{x_0, x_1\} \rightarrow W$ and a first-order formula ψ using at most the individual variables $\{x_0, x_1\}$, we write $\mathcal{M}, v \models \psi$ to denote that ψ holds true in $\mathcal{M}$ under the valuation v (we use here the standard definition). We write $v' = v[\alpha \leftarrow w]$ to denote the valuation obtained from v by putting $v'(\beta) = v(\beta)$ and $v'(\alpha) = w$.

The following, rather technical, lemma states roughly the following: suppose we have a first-order model $\langle W, V \rangle$ containing some point $w \in W$, such that in every point v that is reachable from w through a path labelled by a finite word accepted by the automaton $\mathcal{A}$, the formula $\varphi(\alpha)$ is true, then we can extend V in such a way, that the new model $\langle W, V' \rangle$ will satisfy the translation $t_{\mathcal{A}}(\alpha, \varphi)$ in w.

Lemma 3.2. Let $\mathcal{A}$ be an NDFA and $\varphi(\alpha)$ be a first-order formula with one free variable α. Let $\mathcal{M} = \langle W, V \rangle$ be a first-order structure not interpreting any of the fresh symbols introduced by $t_{\mathcal{A}}(\alpha, \varphi)$ (those of the form $\mathbf{q}$ for every state q of $\mathcal{A}$). Then there is an extension $\mathcal{M}' = \langle W, V' \rangle$ of $\mathcal{M}$ such that, for every $w \in W$, $(\star)$ below is satisfied:

($\star$) For every word $b_1 \cdots b_n \in \Sigma^*$ that is accepted by $\mathcal{A}$, for every sequence $w_1, \ldots, w_n$ of elements of W, such that

$$\langle w, w_1\rangle \in V(\mathbf{R}_{b_1}),\ \langle w_1, w_2\rangle \in V(\mathbf{R}_{b_2}),\ \ldots,\ \langle w_{n-1}, w_n\rangle \in V(\mathbf{R}_{b_n}),$$

we have $\mathcal{M}, v[\alpha \leftarrow w_n] \models \varphi(\alpha)$,

we have

$$\mathcal{M}',\ v[\alpha \leftarrow w] \models t_{\mathcal{A}}(\alpha, \varphi).$$

Proof. Write $\mathcal{A} = \langle Q, s, F, \delta\rangle$. We extend V to also interpret the symbols $\mathbf{q}$, and also we do this in a way that is consistent with the runs of $\mathcal{A}$. For all $w \in W$ and states q of $\mathcal{A}$, we define $w \in V'(\mathbf{q}) \overset{\text{def}}{\Leftrightarrow}$

– for every word $b_1 \cdots b_n \in \mathrm{L}(\mathcal{A})$ and for every sequence $w_1, \ldots, w_n$ of elements of W, such that

$$\langle w, w_1\rangle \in V(\mathbf{R}_{b_1}),\ \langle w_1, w_2\rangle \in V(\mathbf{R}_{b_2}),\ \ldots,\ \langle w_{n-1}, w_n\rangle \in V(\mathbf{R}_{b_n}),$$

we have $\mathcal{M}, v[\alpha \leftarrow w_n] \models \varphi(\alpha)$.

It is easy to check (but tedious to write out because of the size of the statements involved) that for every $w \in W$ satisfying the condition ($\star$), we have

– for the initial state s,

$$\mathcal{M}', v[\alpha \leftarrow w] \models \mathbf{s}(\alpha).$$

– for every transition $\langle q, a, r\rangle \in \delta$,

$$\mathcal{M}' \models \forall\alpha\beta\ [\ \mathbf{R}_a(\alpha, \beta) \Rightarrow (\mathbf{q}(\alpha) \Rightarrow \mathbf{r}(\beta))\ ].$$

– for every ϵ-transition $\langle q, \epsilon, r\rangle \in \delta$,

$$\mathcal{M}' \models \forall\alpha\ [\ \mathbf{q}(\alpha) \rightarrow \mathbf{r}(\alpha)\ ].$$

– for each accepting state $q \in F$,

$$\mathcal{M}' \models \forall\alpha\ [\ \mathbf{q}(\alpha) \rightarrow \varphi(\alpha)\ ].$$

$\mathcal{M}'$ and $\mathcal{M}$ agree on all formulas that do not contain any symbols introduced by $t_{\mathcal{A}}(\alpha, \varphi)$, i.e. those of the form $\mathbf{q}$ for some state q of $\mathcal{A}$. □

Next follows the main theorem about satisfiability preservation.

Theorem 3.3. Let $\phi \in \mathrm{ML}^{\Sigma}$ be a modal formula in NNF. Then the following are equivalent:

1. There exist a $\langle \Sigma, \bar{\cdot} \rangle$-model $\mathcal{M} = \langle W, R, V \rangle$ and a $w \in W$ such that $\mathcal{M}$ satisfies S and $\mathcal{M}, w \models \phi$.
2. $T_{\mathrm{S}}(\phi) \wedge \mathrm{CONV}_{\Sigma}$ is satisfiable in FOL.

Proof. We first prove that **(1)** implies **(2)**. Assume that there exists a $\langle \Sigma, \bar{\cdot} \rangle$-model $\mathcal{M} = \langle W, R, V \rangle$ with a $w \in W$ such that $\mathcal{M}, w \models \phi$ and $\langle W, R \rangle$ satisfies S. We need to construct a model $\mathcal{M}'$ of $T_{\mathrm{S}}(\phi) \wedge \mathrm{CONV}_{\Sigma}$.

In order to do this, we first construct an incomplete interpretation $\mathcal{M}_0 = \langle W, V_0 \rangle$, which will be completed through successive applications of Lemma 3.2. V_0 is obtained as follows:

– For every $a \in \Sigma$, $V_0(\mathbf{R}_a) \stackrel{\mathrm{def}}{=} R_a$,

– For every propositional variable p occurring in ϕ, we set $V_0(\mathbf{p}) \stackrel{\mathrm{def}}{=} V(\mathrm{p})$.

We now have a model interpreting the symbols introduced by $t(\psi, \alpha, \beta)$, but not the symbols introduced by $t_{\mathcal{A}}(\alpha, \psi)$. It is easily checked that CONV_{Σ} holds true in $\langle W, V_0 \rangle$. In order to complete the model construction, we order the $\Box$-subformulae of ϕ in a sequence $[a_1]\psi_1, \ldots, [a_n]\psi_n$ such that every $\Box$-subformula is preceeded by all its $\Box$-subformulae. Hence, $i < j$ implies that $[a_j]\psi_j$ is not a subformula of $[a_i]\psi_i$. Then we iterate the following construction ($1 \leq i \leq n$):

– $\mathcal{M}_i = \langle W, V_i \rangle$ is obtained from $\mathcal{M}_{i-1} = \langle W, V_{i-1} \rangle$ by applying the construction of Lemma 3.2 on $\mathcal{A}_{a_i}$ and $t(\psi_i, \alpha, \beta)$.

Then $\mathcal{M}_n = \langle W, V_n \rangle$ is our final model. Roughly speaking, V_i is equal to V_{i-1} extended with the unary predicate symbols of the form $\mathbf{q}_{\psi_i}$ with q a state of $\mathcal{A}_{a_i}$. The values of the other predicate symbols remain unchanged. We have

– for every $a \in \Sigma$, $V_0(\mathbf{R}_a) = \cdots = V_n(\mathbf{R}_a)$,

– for every propositional variable p of ϕ, $V_0(\mathbf{p}) = \cdots = V_n(\mathbf{p})$.

Additionally,

– for every $j \in \{1, \ldots, n\}$, for every state q of $\mathcal{A}_{a_j}$, $V_j(\mathbf{q}_{\psi_j}) = V_{j+1}(\mathbf{q}_{\psi_j}) = \ldots = V_n(\mathbf{q}_{\psi_j})$.

We show by induction that for every subformula ψ of ϕ, for every $x \in W$, for every valuation v, $\mathcal{M}, x \models \psi$ implies $\mathcal{M}_n, v[\alpha \leftarrow x] \models t(\psi, \alpha, \beta)$. We treat only the modal cases, because the propositional cases are trivial.

- If ψ has form $[a]\psi'$ with $a \in \Sigma$, then $t([a]\psi', \alpha, \beta) = t_{\mathcal{A}_a}(\alpha, t(\psi', \alpha, \beta))$.

  For every word $b_1 \cdots b_k$ accepted by $\mathcal{A}_a$, for every sequence $w_1, \dots, w_k \in W_n$ such that

  $$\langle x, w_1\rangle \in V_n(\mathbf{R}_{b_1}),\ \langle w_1, w_2\rangle \in V_n(\mathbf{R}_{b_2}),\ \dots\ , \langle w_{k-1}, w_k\rangle \in V_n(\mathbf{R}_{b_k}),$$

  also

  $$\langle x, w_1\rangle \in R_{b_1},\ \langle w_1, w_2\rangle \in R_{b_2},\ \cdots\ , \langle w_{k-1}, w_k\rangle \in R_{b_k},$$

  by construction of $V_0, V_1, \cdots, V_n$. Because $\mathcal{M}$ satisfies S, by Lemma 2.4, we also have $\langle x, w_k\rangle \in R_a$, which in turn implies $\langle x, w_k\rangle \in V_n(\mathbf{R}_a)$, by construction of the V_i. Therefore, we have $\mathcal{M}, w_k \models \psi'$. By the induction hypothesis, we have $\mathcal{M}_n, v[\beta \leftarrow w_k] \models t(\psi', \beta, \alpha)$. Let n' be the position of ψ' in the enumeration of $\Box$-subformulae $[a_1]\psi_1, \dots, [a_n]\psi_n$. It is easily checked that

  $$\mathcal{M}_{n'}, v[\beta \leftarrow w_k] \models t(\psi', \beta, \alpha).$$

  Now we have all ingredients of Lemma 3.2 complete, and it follows that

  $$\mathcal{M}_{n'}, v[\alpha \leftarrow x] \models t_{\mathcal{A}_a}(\alpha, t(\psi', \alpha, \beta)).$$

  Since $\mathcal{M}_n$ is a conservative extension $\mathcal{M}_{n'}$, we also get

  $$\mathcal{M}_n, v[\alpha \leftarrow x] \models t_{\mathcal{A}_a}(\alpha, t(\psi', \alpha, \beta)).$$

- If ψ has form $\langle a\rangle\psi'$, then there is a y such that $\langle x, y\rangle \in R_a$ and $\mathcal{M}, y \models \psi'$. By definition of V_0, we have $\langle x, y\rangle \in V_0(\mathbf{R}_a)$ and therefore also $\langle x, y\rangle \in V_n(\mathbf{R}_a)$. By the induction hypothesis, $\mathcal{M}_n, v[\beta \leftarrow y] \models t(\psi', \beta, \alpha)$. Hence,

  $$\mathcal{M}_n, v[\alpha \leftarrow x] \models \exists\beta\ [\ \mathbf{R}_a(\alpha, \beta) \wedge t(\psi', \beta, \alpha)].$$

Next we show that **(2)** implies **(1)**. Assume that $T_{\mathrm{S}}(\phi) \wedge \mathrm{CONV}_\Sigma$ is FOL-satisfiable. This means that there exist a FOL model $\mathcal{M} = \langle W, V\rangle$ and a valuation v such that $\mathcal{M}, v \models T_{\mathrm{S}}(\phi) \wedge \mathrm{CONV}_\Sigma$. We construct a model $\mathcal{M}'$ of ϕ in two stages. First we construct $\mathcal{M}'' = \langle W'', R'', V''\rangle$ as follows.

- $W'' \overset{\text{def}}{=} W$.
- For every $a \in \Sigma$, $R''_a \overset{\text{def}}{=} V(\mathbf{R}_a)$.
- For every propositional variable p, $V''(\mathrm{p}) \overset{\text{def}}{=} V(\mathbf{p})$.

Then define $\mathcal{M}' = \langle W', R', V' \rangle$ where R' is defined from $\langle W', R' \rangle = C_{\mathrm{S}}(\langle W'', R'' \rangle)$ and $V' = V''$. Here C_{S} is the closure operator, defined in Definition 2.18. Intuitively, we construct $\mathcal{M}'$ by copying W and the interpretation of the accessibility relations from $\mathcal{M}$, and applying C_{S} on it. The constructions imply that $W' = W$. Because $\mathcal{M} \models \mathrm{CONV}_{\Sigma}$, the frame $\mathcal{M}''$ is a $\langle \Sigma, \bar{\cdot} \rangle$-frame. By definition of C_{S}, the structure $\mathcal{M}'$ is an S-model, and also a $\langle \Sigma, \bar{\cdot} \rangle$-frame. We now show by induction that for every subformula ψ of ϕ, $\mathcal{M}, v \models t(\psi, \alpha, \beta)$ implies $\mathcal{M}', v(\alpha) \models \psi$.

- If ψ has form $\langle a \rangle \psi'$, then $\mathcal{M}, v \models t(\langle a \rangle \psi', \alpha, \beta)$, that is $\mathcal{M}, v \models \exists \beta\ [\ \mathbf{R}_a(\alpha, \beta) \wedge t(\psi', \beta, \alpha)\ ]$.

  This means there is a $y \in W$, such that $\langle v(\alpha), y \rangle \in V(\mathbf{R}_a)$ and
  $$\mathcal{M}, v[\beta \leftarrow y] \models t(\psi', \beta, \alpha).$$
  By the induction hypothesis, $\mathcal{M}', y \models \psi'$. It follows from the definition of R', using the fact that C_S is increasing (by its definition), that $\langle x, y \rangle \in R'_a$, so we have $\mathcal{M}', x \models \langle a \rangle \psi$.

- If ψ has form $[a]\psi'$, then assume that $\mathcal{M}, v \models t_{\mathcal{A}_a}(\alpha, t(\psi', \alpha, \beta))$. First, we show that for every word $b_1 \cdots b_k$ accepted by $\mathcal{A}_a$, for every sequence $w_1, \ldots, w_k$ of elements of W, for which it is the case that
  $$\langle v(\alpha), w_1 \rangle \in V(\mathbf{R}_{b_1}),\ \langle w_1, w_2 \rangle \in V(\mathbf{R}_{b_2}),\ \ldots, \langle w_{k-1}, w_k \rangle \in V(\mathbf{R}_{b_k}),$$
  the following holds
  $$\mathcal{M}, v[\alpha \leftarrow w_k] \models t(\psi', \alpha, \beta).$$
  Indeed, $\mathcal{M}, v \models \mathbf{s}(\alpha)$, for the initial state s of $\mathcal{A}_a$. It is easy to show by induction on k that the following holds: Let $b_1 \cdots b_k$ be some word over Σ^k. Let q be a state of $\mathcal{A}_a$ such that $\langle s, b_1 \cdot \ldots \cdot b_k, q \rangle \in \delta^*$, for the initial state $s \in Q$. Then for every sequence $w_1, \ldots, w_k$ of elements of W such that
  $$\langle v(\alpha), w_1 \rangle \in V(\mathbf{R}_{b_1}),\ \langle w_1, w_2 \rangle \in V(\mathbf{R}_{b_2}),\ \ldots,\ \langle w_{k-1}, w_k \rangle \in V(\mathbf{R}_{b_k}),$$
  it must be the case that $\mathcal{M}, v[\alpha \leftarrow w_k] \models \mathbf{q}(\alpha)$. Then the result follows from the fact that $\mathcal{M}, v[\alpha \leftarrow w_k] \models \mathbf{q}(\alpha) \Rightarrow \psi'(\alpha)$, for every accepting state q of $\mathcal{A}_a$.

  Now assume that in $\mathcal{M}'$, we have a world y for which $R'_a(x, y)$. Then, by Theorem 2.6, there is a word $u \in \mathrm{L}(\mathcal{A}_a)$ for which $a \Rightarrow^*_{\mathsf{S} \cup \overline{\mathsf{S}}} u$ and $R''_u(x, y)$. By the above property, we have
  $$\mathcal{M}, v[\alpha \leftarrow y] \models t(\psi', \alpha, \beta).$$
  By the induction hypothesis, we obtain $\mathcal{M}', y \models \psi'$.

□

The uniformity of the translation allows us to establish forthcoming Theorem 3.4. We first define the general satisfiability problem for regular grammar logic with converse, denoted by GSP(REG^c), as follows:

input: A finite semi-Thue system S closed under converse, in which either all production rules are left-linear, or all production rules are right-linear, and an ML$^\Sigma$-formula ϕ;

question: is ϕ S-satisfiable?

We need to restrict the form of the semi-Thue system to a form from which the automata $\mathcal{A}_a$ can be computed. Even if one knows that some language L is regular, then there is no effective way of obtaining an NDFA for L. This is a consequence of Theorem 2.12 (iii) in (Rozenberg and Salomaa, 1994).

Theorem 3.4.

(I) The S-satisfiability problem is in EXPTIME for every regular semi-Thue system closed under converse.

(II) GSP(REG^c) is EXPTIME-complete.

Theorem 3.4(I) is a corollary of Theorem 3.3. The lower bound in Theorem 3.4(II) is easily obtained by observing that there exist known regular grammar logics (even without converse) that are already EXPTIME-complete, e.g. K with the universal modality. The upper bound in Theorem 3.4(II) is a consequence of the facts that $T_{\mathrm{S}}(\phi)$ can be computed in logarithmic space in $|\phi| + |\mathrm{S}|$ and the guarded fragment has an EXPTIME-complete satisfiability problem when the arity of the predicate symbols is bounded by some fixed $k \geq 2$ (Grädel, 1999b). We use here the fact that one needs only logarithmic space to build a finite automaton recognizing the language of a right-linear [resp. left-linear] grammar.

Extensions to context-free grammar logics with converse. When S is a context-free semi-Thue system with converse, S-satisfiability can be encoded as for the case of regular semi-Thue systems with converse by adding an argument to the predicate symbols of the form $\mathbf{q}_\psi$. The details are omitted here but we provide the basic intuition. Each language $\mathrm{L}_{\mathrm{S}}(a)$ is context-free and therefore there is a pushdown automaton (PDA) $\mathcal{A}$ recognizing it. The extra argument for the $\mathbf{q}_\psi$s represents the content of the stack and the map $t_{\mathcal{A}}(\alpha, \varphi)$ can be easily extended in the presence of stacks. For instance, the stack content aab can be represented by the first-order term $a(a(b(\epsilon)))$ with the adequate arity for the function symbols a, b, and ϵ. Suppose we have the following transition rule: if the PDA is in state q, the current input symbol is a, and the top symbol of the stack is b_0, then the new state is q' and b_0 is replaced by $b_1 \cdots b_n$ on the top of the stack. This rule is encoded in FOL as follows:

$$\forall\ \alpha, \beta, \gamma,\ (t_a(\alpha, \beta) \Rightarrow (\mathbf{q}(\alpha, b_0(\gamma)) \Rightarrow \mathbf{q}'(\beta, b_1(\ldots b_n(\gamma)\ldots)))).$$

The translation T_{S} is then defined with the context-free version of $t_{\mathcal{A}}(\alpha, \varphi)$. Satisfiability preservation is also guaranteed but the first-order fragment in which the translation is performed (beyond GF) is no longer decidable. Hence, although this provides a new translation of context-free grammar logics with converse, from the point of view of effectivity, this is not better than the relational translation which is also known to be possible when S is a context-free semi-Thue system with converse.

4. The Borders of the Translation Method

In this section we try to answer the following question: given a finite context-free semi-Thue system S closed under converse, how to find out whether the modal logic based on S can be translated by the method of Section 3.1? This is a natural question, because modal logics are usually presented by modal axioms, and in most cases the semi-Thue system naturally corresponds to the modal axioms.

As stated in Sect. 2.2, a logic can be translated if for every letter $a \in \Sigma$, the language $\mathrm{L}_{\mathrm{S}}(a) = \{u \in \Sigma^* \mid a \Rightarrow^*_{\mathrm{S}} u\}$ are regular. This question is in general undecidable, because it is already undecidable whether the language generated by a linear grammar is regular (see e.g. (Mateescu and Salomaa, 1997, page 31)).

However, regularity of the languages $\mathrm{L}_{\mathrm{S}}(a)$ is not a necessary condition for existence of a translation. The reason for this is that different semi-Thue systems may characterize the same logic. Exactly which context-free semi-Thue systems characterize the same logic is determined by Theorem 4.1 below.

Theorem 4.1. Let Σ be an alphabet with converse mapping $\bar{\cdot}$ and $\mathrm{S}_1, \mathrm{S}_2$ be context-free semi-Thue systems closed under converse. If for all $(i_1, i_2) \in \{(1,2), (2,1)\}$ and rules $u \to v \in \mathrm{S}_{i_1}$, there is a string $z \in \Sigma^*$, such that

$$u \Rightarrow^*_{\mathrm{S}_{i_2}} z \text{ and } z \Rightarrow^*_{E_\Sigma} v,$$

then S_1 and S_2 define the same set of $\langle \Sigma, \bar{\cdot} \rangle$-frames.

Proof. It follows from Theorem 2.5 that a $\langle\Sigma, \bar{\cdot}\rangle$-frame satisfies S_1 if and only if it satisfies S_2. $\square$

Theorem 4.1 determines when two context-free semi-Thue systems define the same modal logic. Using this equivalence, the criterion above can be refined to: a modal logic $\mathcal{L}$ based on finite context-free semi-Thue system S closed under converse can be translated by the method of Section 3.1 if (and approximately only-if) there is a finite regular semi-Thue system S' closed under converse which is equivalent to S, for which the languages $\mathrm{L}_{\mathrm{S}'}(a)$ are regular. We do not have a general method for answering this question, and we don't know whether the problem is decidabile.

There exist pairs of context-free semi-Thue systems S_1 and S_2 defining the same logic, where S_1 is non-regular while S_2 is regular. An example of such a pair is given in the following examples:

Example 4.1. The euclideanity condition can be generalized by considering frame conditions of the form $(R_a^{-1})^n; R_a \subseteq R_a$ for some $n \geq 1$. The context-free semi-Thue system corresponding to this inclusion is $\mathrm{S}_n = \{a \to \overline{a}^n a,\ \overline{a} \to \overline{a}a^n\}$. The case $n = 1$ corresponds to euclideanity, which is regular, see Example 2.1 and Example 3.1. We will show that in general, for $n > 1$, the language $\mathrm{L}_{\mathrm{S}_n}(a)$ is not regular. Nevertheless, S_n-satisfiability restricted to formulae with only the modal operator $[a]$ is known to be decidable, see e.g. (Gabbay, 1975; Hustadt and Schmidt, 2003). To see why the languages $\mathrm{L}_{\mathrm{S}_n}(a)$ are not regular, consider strings of the following form:

$$\sigma_n(i_1, i_2) = (\overline{a}a^{n-1})^{i_1}\ a\ (\overline{a}^{n-1}a)^{i_2}.$$

$$\overline{\sigma}_n(i_1, i_2) = (\overline{a}a^{n-1})^{i_1}\ \overline{a}\ (\overline{a}^{n-1}a)^{i_2}.$$

We show that

$$(\ a \Rightarrow^*_{\mathrm{S}_n} \sigma_n(i_1, i_2) \text{ and } a \Rightarrow^*_{\mathrm{S}_n} \overline{\sigma}_n(i_1, i_2 + 1)\ ) \text{ iff } i_1 = i_2.$$

In order to check that the equivalence holds from right to left, observe that $a = \sigma_n(0, 0)$, and

$$\sigma_n(0, 0) \Rightarrow_{\mathrm{S}_n} \overline{\sigma}_n(0, 1) \Rightarrow_{\mathrm{S}_n} \sigma_n(1, 1) \Rightarrow_{\mathrm{S}_n} \cdots$$

$$\Rightarrow_{\mathrm{S}_n} \sigma_n(i, i) \Rightarrow_{\mathrm{S}_n} \overline{\sigma}_n(i, i + 1) \Rightarrow_{\mathrm{S}_n} \sigma_n(i + 1, i + 1) \Rightarrow_{\mathrm{S}_n} \cdots$$

We now prove the equivalence from left to right. Let us say that u is *a predecessor of* v if $u \Rightarrow_{\mathrm{S}_n} v$. Then it is sufficient to observe the following:

1. A string of form $\sigma_n(0, j)$ with $j > 0$ has no predecessor.

2. A string of form $\sigma_n(i+1,j)$ has only one predecessor, namely $\overline{\sigma}_n(i,j)$.
3. A string of form $\overline{\sigma}_n(i,0)$ with $i \geq 0$ has no predecessor.
4. A string of form $\overline{\sigma}_n(i,j+1)$ has only one predecessor, namely $\sigma_n(i,j)$.

To have a predecessor, a string must have a sequence of at least n consecutive a's or $\overline{a}$'s. The strings of the form either 1 or 3 have no such sequence. The strings of the form either 2 or 4 have exactly one such sequence.
We have

$$\mathrm{L}_{\mathrm{S}_n}(a) \cap \{\sigma_n(i,j) \mid i \geq 0,\ j \geq 0\} = \{\sigma_n(i,i) \mid i \geq 0\}.$$

The language $\{\sigma_n(i,i) \mid i \geq 0\}$ is clearly not regular (we assume $n > 1$) and $\{\sigma_n(i,j) \mid i \geq 0,\ j \geq 0\}$ is clearly regular. Since the regular languages are closed under intersection, $\mathrm{L}_{\mathrm{S}_n}(a)$ cannot be regular for $n > 1$.
$\lhd$

We will now show that, although

$$\mathrm{S}_2 = \{\ a \rightarrow \overline{a}\overline{a}a, \quad \overline{a} \rightarrow \overline{a}aa\ \}$$

is not regular, the logic defined by it, is regular. The reason for this is the fact that S_2 defines the same logic as

$$\mathrm{S}_2' = \{\ a \rightarrow \overline{a}\overline{a}a, \quad a \rightarrow \overline{a}aa, \quad \overline{a} \rightarrow \overline{a}\overline{a}a, \quad \overline{a} \rightarrow \overline{a}aa\ \},$$

which is regular. In order to show that S_2 and S_2' define the same logic, it suffices to observe that $\mathrm{S}_2 \subseteq \mathrm{S}_2'$, and that

$$a \Rightarrow_{\mathrm{S}_2} \overline{a}\overline{a}a \Rightarrow_{\mathrm{S}_2} (\overline{a}aa)\overline{a}a \Rightarrow_{E_{\{a,\overline{a}\}}} \overline{a}a(a), \text{ and}$$

$$\overline{a} \Rightarrow_{\mathrm{S}_2} \overline{a}aa \Rightarrow_{\mathrm{S}_2} \overline{a}a(\overline{a}\overline{a}a) \Rightarrow_{E_{\{a,\overline{a}\}}} (\overline{a})\overline{a}a.$$

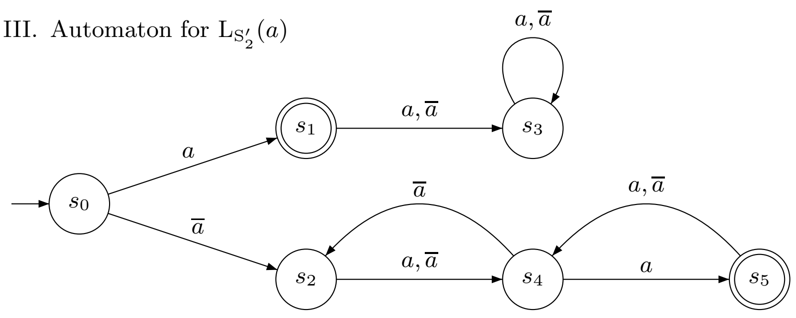
In order to show that S_2' is regular, we show that $\mathrm{L}_{\mathrm{S}_2'}(a)$ is recognized by the automaton of Table III.

Lemma 4.2. The automaton in Table III recognizes the language $\mathrm{L}_{\mathrm{S}_2'}(a)$.

Proof. It is clear that state s_1 accepts only the word a. Every run ending in s_5 consists of three parts:

1. The path (s_0, s_2, s_4), accepting either $\overline{a}a$ or $\overline{a}\overline{a}$.

Table III. Automaton for $\mathrm{L}_{\mathrm{S}'_2}(a)$



2. A number of cycles of form (s_4, s_5, s_4), or (s_4, s_2, s_4) accepting aa, $a\overline{a}$, $\overline{a}a$, or $\overline{a}\overline{a}$. These are all possible words of length two.
3. The path (s_4, s_5), accepting the word a.

As a consequence, a word u is accepted by the automaton iff it has one of the following three forms:

$$a, \quad \overline{a}a\Sigma^{2i}a, \quad \text{or } \overline{a}\overline{a}\Sigma^{2i}a, \tag{4}$$

where Σ^{2i} is an arbitrary word of length $2i$ on the alphabet $\Sigma = \{a, \overline{a}\}$.

We first show that every string in $\mathrm{L}_{\mathrm{S}'_2}(a)$ has the form 4. The language $\mathrm{L}_{\mathrm{S}'_2}(a)$ is inductively defined as the smallest set containing a and closed under rewriting by rules of S'_2. Therefore, it is sufficient to show that a has form 4 and that form 4 is preserved by rewriting under rules in S'_2.

If u is of form $\overline{a}a\Sigma^{2i}a$, rewriting at $\overline{a}$ results in $(\overline{a}\overline{a}a)a\Sigma^{2i}a$ or $(\overline{a}aa)a\Sigma^{2i}a$, which is of the form $\overline{a}\overline{a}\Sigma^{2i+1}a$ or $\overline{a}a\Sigma^{2i+1}a$. Rewriting at the first a results either in $\overline{a}(\overline{a}\overline{a}a)\Sigma^{2i}a$ or $\overline{a}(\overline{a}aa)\Sigma^{2i}a$. Both can be written as $\overline{a}\overline{a}\Sigma^{2i+1}a$. Rewriting at the last a results either in $\overline{a}a\Sigma^{2i}\overline{a}\overline{a}a$ or $\overline{a}a\Sigma^{2i}\overline{a}aa$, which both can be written as $\overline{a}a\Sigma^{2i+1}a$. Rewriting in Σ^{2i} results also in a string of one of the three forms. The 7 possible rewrites in $\overline{a}\overline{a}\Sigma^{2i}a$ can be analyzed analogously.
Next we show by induction on i that every word of the form 4 is in $\mathrm{L}_{\mathrm{S}'_2}(a)$. For the base case $i = 0$, it is immediate that a, $\overline{a}aa$, $\overline{a}\overline{a}a \in \mathrm{L}_{\mathrm{S}'_2}(a)$. Now assume that, for some i, every word of the form $\overline{a}a\Sigma^{2i}a$ or $\overline{a}\overline{a}\Sigma^{2i}a$ belongs to $\mathrm{L}_{\mathrm{S}'_2}(a)$. We show that every word of the form $\overline{a}a\Sigma^{2(i+1)}a$ or $\overline{a}\overline{a}\Sigma^{2(i+1)}a$ also belongs to $\mathrm{L}_{\mathrm{S}'_2}(a)$. In each case, we find a string of form 4 (but with parameter i) from which the current string can be obtained using a single rewrite step by a rule in S'_2. If one

expands the first two letters of Σ^{2i+2}, one obtains the following forms:

$$\overline{a}a(aa)\Sigma^{2i}a,\ \ \overline{a}a(a\overline{a})\Sigma^{2i}a,\ \ \overline{a}a(\overline{a}a)\Sigma^{2i}a,\ \ \overline{a}a(\overline{a}\overline{a})\Sigma^{2i}a,\ \text{and}$$
$$\overline{a}\overline{a}(aa)\Sigma^{2i}a,\ \ \overline{a}\overline{a}(a\overline{a})\Sigma^{2i}a,\ \ \overline{a}\overline{a}(\overline{a}a)\Sigma^{2i}a,\ \ \overline{a}\overline{a}(\overline{a}\overline{a})\Sigma^{2i}a.$$

Case 1.
By induction hypothesis, $\overline{a}a\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$ and

$$\overline{a}a\Sigma^{2i}a \Rightarrow_{\{\overline{a}\to\overline{a}aa\}} \overline{a}a(aa)\Sigma^{2i}a$$

by rewriting at the first position. Hence, $\overline{a}a(aa)\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$.
Case 2.
By induction hypothesis, $\overline{a}\overline{a}\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$ and

$$\overline{a}\overline{a}\Sigma^{2i}a \Rightarrow_{\{\overline{a}\to\overline{a}aa\}} \overline{a}a(a\overline{a})\Sigma^{2i}a$$

by rewriting at the first position. Hence, $\overline{a}a(a\overline{a})\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$.
Cases 5 and 6. Similar to the cases 5 and 6 using the rule $\overline{a} \to \overline{a}\overline{a}a$.
Case 7.
By induction hypothesis, $\overline{a}\overline{a}\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$ and

$$\overline{a}\overline{a}\Sigma^{2i}a \Rightarrow_{\{\overline{a}\to\overline{a}\overline{a}a\}} \overline{a}\overline{a}(\overline{a}a)\Sigma^{2i}a$$

by rewriting at the second position. Hence, $\overline{a}\overline{a}(\overline{a}a)\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$.
Cases 4 and 8.
In order to treat both cases, we pose u to denote a string in $\{\overline{a}a, \overline{a}\overline{a}\}$.
By induction hypothesis, $u\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$ and

$$u\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a) \Rightarrow_{\{a\to\overline{a}\overline{a}a\}} u(\overline{a}\overline{a})\Sigma^{2i}a$$

by rewriting at the first occurrence of a in $\Sigma^{2i}a$. Hence, $u(\overline{a}\overline{a})\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$.
Case 3.
By induction hypothesis, $\overline{a}a\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$. Write $\Sigma^{2i} = (\overline{a}a)^{i_1}\Sigma^{2i_2}$, where $i = i_1 + i_2$, and i_1 is maximal.
Case 3.1.: $\Sigma^{2i_2}a$ starts by a.
We have

$$\overline{a}a\Sigma^{2i}a \Rightarrow_{\{a\to\overline{a}aa\}} \overline{a}a(\overline{a}a)\Sigma^{2i}a$$

by rewriting at the first occurrence of a in $\Sigma^{2i_2}a$.
Case 3.2: Σ^{2i_2} starts with $\overline{a}\overline{a}$.
$\Sigma^{2i_2}a$ is of the form $\overline{a}^k au$ for some $k \geq 2$ and string u. Since $\overline{a}^{k-2}au \Rightarrow_{\{a\to\overline{a}\overline{a}a\}}$ $\overline{a}^k au$ and $(\overline{a}a)^{i_1}\overline{a}^{k-2}au \in \mathrm{L}_{\mathrm{S}_2'}(a)$ by induction hypothesis, we have that $\overline{a}a\overline{a}a\Sigma^{2i}a \in \mathrm{L}_{\mathrm{S}_2'}(a)$.

□

The regularity of S_2' and its equivalence to S_2 make it possible to deduce the following upper bound.

Theorem 4.3. The bimodal logic whose set of frames is the set of frames satisfying S_2 is a regular grammar logic with converse. Hence, its satisfiability problem can be solved in EXPTIME.

The automaton in Table III was discovered by a computer program. Given a language L over an alphabet Σ, one can define the following equivalence relation $\equiv_{\mathrm{L}}$ on strings over Σ : For two words $u_1, u_2 \in \Sigma^*$, $u_1 \equiv_{\mathrm{L}} u_2$ iff for all $v \in \Sigma^*$, $u_1{\cdot}v \in \mathrm{L} \Leftrightarrow u_2{\cdot}v \in \mathrm{L}$. By the Myhill-Nerode theorem, if $\equiv_{\mathrm{L}}$ partitions Σ^* into a finite set of equivalence-classes, then L is regular, and the equivalence classes define the states of the minimal DFA recognizing L. If one tries sufficiently many v's, one has a high chance of finding the right equivalence classes. Of course one cannot be certain in general that the automata returned by the program are correct, but in most cases verifying an automaton is easier than finding one. The same computer program has also proposed an automaton for the modal logic defined by $\mathrm{S}_3 = \{a \to \overline{a}^3 a, \quad \overline{a} \to \overline{a} a^3\}$. This automaton has 19 states. Based on the fact that the grammar logics defined by $\mathrm{S}_1, \mathrm{S}_2$ and S_3 can be characterized by a regular language, we conjecture that all of the grammar logics defined by a grammar of form S_i are regular. At the same time, it appears that the modal logics defined by the context-free semi-Thue systems $\hat{\mathrm{S}}_i = \{a \to a^i \overline{a}, \quad \overline{a} \to a \overline{a}^i\}$ with $i > 1$, are non-regular, based on the output of the same computer program. We conjecture that the output of the program is correct and that the grammar logics defined by $\hat{\mathrm{S}}_i,\ i > 1$ are indeed non-regular.

5. Related work

In this section, we compare our contribution to translations of modal logics similar to ours, to tableaux-based proof systems with single steps and to the characterization of star-free languages with first-order logic over finite words. Before doing so, let us mention some other relevant works.

Complexity issues for regular grammar logics have been studied in (Demri, 2001; Demri, 2002) (see also (Baldoni, 1998; Baldoni et al., 1998)) whereas grammar logics are introduced in (del Cerro and Penttonen, 1988). Frame conditions involving the converse relations are not treated in (Demri, 2001; Demri, 2002). These are needed for example for S5 modal connectives. The current work can be viewed as a natural continuation of (de Nivelle, 1999) and (Demri, 2001). Translation of regular grammar logics into converse PDL can be found in the preprint (Demri and de Nivelle, 2004, Sect. 4) extending (Demri, 2001).

The frame conditions considered in the present work can be defined by the MSO definable closure operators (Ganzinger et al., 1999). However, it is worth noting that by contrast to what is done in (Ganzinger et al., 1999), we obtain the optimal complexity upper bound for the class of regular grammar logics with converse (EXPTIME) since the first-order fragment we consider is much more restricted than the one in (Ganzinger et al., 1999). Moreover, we do not use MSO definable built-in relations, just plain GF^2.

5.1. Incorporating a theory in the translation

Unlike the standard relational translation from modal logic into classical predicate logic (see e.g., (Fine, 1975; van Benthem, 1976; Morgan, 1976; Moore, 1977)), the subformulae in $T_{\mathrm{S}}(\phi)$ mix the frame conditions and the interpretation of the logical connectives. Frame conditions are incorporated in our translation as done also in (Schmidt and Hustadt, 2004). Such a feature is shared by many other translations dealing for modal logics, see e.g. (Balbiani and Herzig, 1994; Demri and Goré, 2002). However, the work (Schmidt and Hustadt, 2004) is closely related to ours. Probably the main similarities are the following ones.

- Both translations are from a large class of modal logics into GF^2.
- Translations of the modal logics K,T,K4 and S4 in (Schmidt and Hustadt, 2004) and in this paper are essentially the same, once minor differences are disgarded (NNF is our work and renaming in (Schmidt and Hustadt, 2004)). For example, the clause schema for the K4 axiom $[a]\mathrm{p} \Rightarrow [a][a]\mathrm{p}$ in (Schmidt and Hustadt, 2004) is the following:

  $$\forall\ x\ Q_{\Box \mathrm{p}}(x) \Rightarrow (\forall\ y\ \mathbf{R}_a(x,y) \Rightarrow Q_{\Box \mathrm{p}}(y)).$$

  This clause schema is obtained from $\Box\mathrm{p} \Rightarrow \Box\Box\mathrm{p}$ by performing a partial translation that stops before the innermost modalities are eliminated and by renaming subformulae. In this paper, the letter a from a K4 modal operator $[a]$ is related to the regular language a^+ that can be recognized by the finite-state automaton $\mathcal{A}_0 = \langle \{q_0, q_1\}, q_0, q_1, \{q_0 \xrightarrow{a} q_1, q_1 \xrightarrow{a} q_1\}\rangle$. The formula $t_{\mathcal{A}_0}(\alpha, \varphi(\alpha))$ introduced in Definition 3.2 contains a conjunct of the form

  $$\forall\ x\ \mathbf{q_1}(x) \Rightarrow (\forall\ y\ \mathbf{R}_a(x,y) \Rightarrow \mathbf{q_1}(y)),$$

  where $\mathbf{q_1}$ is also a monadic predicate symbol depending on $\varphi(\alpha)$ which is after all nothing else than a predicate symbol of the above form $Q_{\Box \mathrm{p}}$ depending on p.
- Both methods require some preliminary knowledge. In our case, the grammar logic at hand needs to be shown regular whereas

in (Schmidt and Hustadt, 2004) one needs to determine how many finite instances of the clause schemata obtained from axioms are sufficient for completeness. Both problems are difficult in general but for many known logics the problem can be solved.

5.2. Relationships with first-order logic over finite words.

The method of translating finite automata into first-order formulas by introducing unary predicate symbols for the states, is reminiscent to the characterization of regular languages in terms of Monadic Second-Order Logic over finite words, namely SOM[+1], see e.g. (Straubing, 1994). Similarly, the class of languages with a *finite* syntactic monoid is precisely the class of regular languages. Our encoding into GF^2 is quite specific since

- we translate into an EXPTIME fragment of FOL, namely GF^2, neither into full FOL nor into a logic over finite words;
- we do not encode regular languages into GF^2 but rather modal logics whose frame conditions satisfy some regularity conditions, expressible in GF with built-in relations (Ganzinger et al., 1999);
- not every regularity condition can be encoded by our method since we require a closure condition.

Hence, the similarity between the encoding of regular languages into SOM[+1] and our translation is quite superficial. The following argument provides some more evidence that the similarity exists only at the syntactic level. The class of regular languages definable with the first-order theory of SOM[+1] is known as the class of star-free languages (their syntactic monoids are finite and aperiodic), see e.g. (Perrin, 1990). However, the regular language $\mathrm{L}_{\mathrm{S}}(a) = (b \cdot b)^*(a \cup \epsilon)$ obtained with the regular semi-Thue system $\mathrm{S} = \{a \rightarrow bba,\ a \rightarrow \epsilon\}$ produces a regular grammar logic with converse that can be translated into GF^2 by our method. Observe that the language $(b \cdot b)^*(a \cup \epsilon)$ is not star-free, see e.g. (Pin, 1994). By contrast, $(a \cdot b)^*$ is star-free but it is not difficult to show that there is no context-free semi-Thue system S such that $\mathrm{L}_{\mathrm{S}}(a) = (a \cdot b)^*$ since a is not in $(a \cdot b)^*$. As a conclusion, our translation into GF^2 is based on principles different from those between star-free regular languages and first-order logic on finite words. Other problems on (tree) automata translatable into classical logic can be found in (Verma, 2003).

6. Concluding Remarks

The two main contributions of the paper are the following:

– for every regular grammar logic with converse the design of a logspace translation into GF^2,

– to characterize when two regular grammar logics define the same set of satisfiable formulae.

As a by-product, our work allows us to answer positively to some questions left open in (Demri, 2001). Typically, we provide evidence that the first-order fragment to translate the regular grammar logics with converse into is simply GF^2: there is no need for first-order fragment augmented with fixed-point operators, as far as regular grammar logics are concerned.

In our view, Theorem 2.5 can be interpreted as confirming that the use of grammars for defining modal logics, is natural. Theorem 2.5 completely determines the behaviour of grammar rules on frames in terms of the behaviour of grammar rules on words. We end by listing a few open problems that we believe are worth investigating.

1. The study of the computational behaviour of the translation to mechanize modal logics using for instance (de Nivelle and Pratt-Hartmann, 2001) should be further investigated.
2. Although regular grammar logics (with converse) can be viewed as fragments of propositional dynamic logic (see e.g. (Demri and de Nivelle, 2004, Sect. 4)), it remains open whether the full PDL can be translated into GF^2 with a similar, almost-structure preserving transformation. We know that there exists a logarithmic space transformation, but we do not want to use first principles on Turing machines.
3. Is there a PSPACE fragment of GF^2 in which the following modal logics can be naturally embedded: S4, $S4_t$ (S4 with past-time operators), Grz, and G? (to quote a few modal logics in PSPACE, see e.g. (Chagrov and Zakharyaschev, 1997)).
4. Can our translation method be extended to a reasonable fragment of first-order modal logics?
5. Combining the translation from S4 into GF^2 with Gödel's translation from intuitionistic logic into S4 (see (Troelstra and Schwichtenberg, 1996)), one obtains a translation from intuitionistic logic into GF^2, see e.g. the preprint (Demri and de Nivelle, 2004, Sect. 5). Can it be extended to a reasonable fragment of first-order intuitionistic logic?

6. Are the conjectures at the end of Section 4 true?

Acknowledgements

The authors would like to thank the referees for many valuable remarks and suggestions greatly improving the readability of this document.

References

Abadi, M., M. Burrows, B. Lampson, and G. Plotkin: 1993, 'A calculus for access control in distributed systems'. *ACM Transactions on Prog. Languages and Systems* **15**(4), 706–734.

Alechina, N. and D. Shkatov: 2003, 'On decidability of intuitionistic modal logics'. In: *Third Workshop on Methods for Modalities, Nancy.*

Andreka, H., I. Nemeti, and J. van Benthem: 1998, 'Modal languages and Bounded Fragments of Predicate Logic'. *Journal of Philosophical Logic* **27**(3), 217–274.

Areces, C., P. Blackburn, and M. Marx: 2000, 'Complexity of Hybrid Temporal Logics'. *Logic Journal of the IGPL* **8**(5), 653–679.

Balbiani, P. and A. Herzig: 1994, 'A translation from the modal logic of provability into K4'. *Journal of Applied Non-Classical Logics* **4**, 73–77.

Baldoni, M.: 1998, 'Normal Multimodal Logics: Automated Deduction and Logic Programming'. Ph.D. thesis, Università degli Studi di Torino.

Baldoni, M., L. Giordano, and A. Martelli: 1998, 'A Tableau Calculus for Multimodal Logics and Some (Un)Decidability results'. In: H. de Swart (ed.): *TABLEAUX-8*, Vol. 1397 of *Lecture Notes in Artificial Intelligence*. pp. 44–59, Springer.

Blackburn, P. and M. Marx: 2002, 'Remarks in Gregory's "Actually" operator'. *Journal of Philosophical Logic* **31**(1), 281–288.

Caucal, D.: 1996, 'On infinite transition graphs having a decidable monadic theory'. In: *ICALP'96*, Vol. 1099 of *Lecture Notes in Computer Science*. pp. 194–205, Springer, Berlin.

Caucal, D.: 2003, 'On infinite transition graphs having a decidable monadic theory'. *Theoretical Computer Science* **290**, 79–115.

Chagrov, A. and V. Shehtman: 1994, 'Algorithmic aspects of propositional tense logics'. In: L. Pacholski and J. Tiuryn (eds.): *CSL-8, Kazimierz, Poland*, Vol. 933 of *Lecture Notes in Computer Science*. pp. 442–455, Springer, Berlin.

Chagrov, A. and M. Zakharyaschev: 1997, *Modal Logic*. Clarendon Press, Oxford.

Courcelle, B.: 1994, 'Monadic second-order definable graph transductions: a survey'. *Theoretical Computer Science* **126**, 53–75.

D'Agostino, G., A. Montanari, and A. Policriti: 1995, 'A set-theoretical translation method for polymodal logics'. *Journal of Automated Reasoning* **15**, 317–337.

de Nivelle, H.: 1998, 'A Resolution Decision Procedure for the Guarded Fragment'. In: C. Kirchner and H. Kirchner (eds.): *CADE-15, Lindau, Germany*, Vol. 1421 of *Lecture Notes in Artificial Intelligence*. pp. 191–204, Springer.

de Nivelle, H.: 1999, 'Translation of S4 and K5 into GF and 2VAR'. Manuscript, available from `http://www.mpi-sb.mpg.de/~nivelle`.

de Nivelle, H.: 2001, 'Translation of S4 and K5 into GF and 2VAR'. Slides available from `http://www.mpi-sb.mpg.de/~nivelle/` on WWW.

de Nivelle, H. and M. de Rijke: 2003, 'Deciding the Guarded Fragments with Resolution'. *Journal of Symbolic Computation* **35**(1), 21–58.

de Nivelle, H. and I. Pratt-Hartmann: 2001, 'A Resolution-Based Decision Procedure for the Two-Variable fragment With Equality'. In: R. Goré, A. Leitsch, and T. Nipkow (eds.): *IJCAR'01*, Vol. 2083 of *Lecture Notes in Computer Science*. pp. 211–225, Springer.

de Nivelle, H., R. Schmidt, and U. Hustadt: 2000, 'Resolution-based methods for modal logics'. *Logic Journal of the IGPL* **8**(3), 265–292.

del Cerro, L. F. and O. Gasquet: 2002, 'A General Framework for Pattern-Driven Modal Tableaux'. *Logic Journal of the IGPL* **10**(1), 51–83.

del Cerro, L. F. and O. Gasquet: 2004, 'Modal tableaux for reasoning about diagrams'. In: A. Pietruszczak and J. Malinowski (eds.): *Essays in Logic and Ontology (Dedicated to Jerzy Perzanowski)*, Poznan Studies in the Philosophy and the Humanities. Amsterdam, The Netherlands: Rodopi, p. To appear.

del Cerro, L. F. and M. Penttonen: 1988, 'Grammar logics'. *Logique et Analyse* **121–122**, 123–134.

Demri, S.: 2000, 'The Nondeterministic Information Logic NIL is PSPACE-complete'. *Fundamenta Informaticae* **42**, 211–234.

Demri, S.: 2001, 'The Complexity of Regularity in Grammar Logics and Related Modal Logics'. *Journal of Logic and Computation* **11**(6), 933–960.

Demri, S.: 2002, 'Modal Logics with Weak Forms of Recursion: PSPACE specimens'. In: M. de Rijke, H. Wansing, F. Wolter, and M. Zakharyaschev (eds.): *Advances in Modal Logics, selected papers from 3rd Workshop on Advances in Modal Logics (AIML'2000), Leipzig, Germany, Oct. 2000*. pp. 113–138, World Scientific.

Demri, S. and H. de Nivelle: 2004, 'Deciding regular grammar logics with converse through first-order logic'. arXiv:cs.LO/0306117.

Demri, S. and R. Goré: 2002, 'Theoremhood Preserving Maps Characterising Cut Elimination for Modal Provability Logics'. *Journal of Logic and Computation* **12**(5), 861–884.

Fagin, R., J. Halpern, Y. Moses, and M. Vardi: 1995, *Reasoning about Knowledge*. The MIT Press.

Fine, K.: 1975, 'Some connections between elementary and modal logic'. In: S. Kanger (ed.): *3rd Scandinavian Logic Symposium*. pp. 15–31, North Holland.

Gabbay, D.: 1975, 'Decidability results in non-classical logics'. *Annals of Mathematical Logic* **8**, 237–295.

Gabbay, D.: 1981, 'Expressive Functional Completeness in Tense Logic'. In: U. Mönnich (ed.): *Aspects of Philosophical Logic*. Reidel, pp. 91–117.

Ganzinger, H. and H. de Nivelle: 1999, 'A Superposition Decision Procedure for the Guarded Fragment with Equality'. In: *LICS'99*. pp. 295–305.

Ganzinger, H., C. Meyer, and M. Veanes: 1999, 'The two-variable guarded fragment with transitive relations (extended abstract)'. In: *LICS'99*. pp. 24–34, IEEE Computer Society Press.

Goranko, V. and S. Passy: 1992, 'Using the universal modality: gains and questions'. *Journal of Logic and Computation* **2**(1), 5–30.

Goré, R.: 1999, 'Tableaux methods for modal and temporal logics'. In: M. d'Agostino, D. Gabbay, R. Hähnle, and J. Posegga (eds.): *Handbook of Tableaux Methods*. pp. 297–396, Kluwer.

Grädel, E.: 1999a, 'Decision procedures for guarded logics'. In: H. Ganzinger (ed.): *CADE'99*, Vol. 1632 of *Lecture Notes in Artificial Intelligence*. pp. 31–51, Springer.

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

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

Grädel, E. and I. Walukiewicz: 1999, 'Guarded Fixed Point Logic'. In: *LICS'99*. pp. 45–54.

Gregory, D.: 2001, 'Completeness and decidability results for some propositional modal logics containing "Actually" operators'. *Journal of Philosophical Logic* **30**(1), 57–78.

Herzig, A.: 1989, 'Raisonnement automatique en logique modale et algorithmes d'unification'. Ph.D. thesis, Université Paul Sabatier, Toulouse.

Heuerding, A., M. Seyfried, and H. Zimmermann: 1996, 'Efficient loop-check for backward proof search in some non-classical propositional logic'. In: P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi (eds.): *Theorem Proving with Analytic Tableaux and Related Methods, 5th International Workshop (TABLEAUX'96)*, Vol. 1071 of *Lecture Notes in Computer Science*. pp. 210–225, Springer, Berlin.

Hirsch, C. and S. Tobies: 2002, 'A Tableau Algorithm for the Clique Guarded Fragment'. In: M. de Rijke, H. Wansing, F. Wolter, and M. Zakharyaschev (eds.): *Advances in Modal Logics, selected papers from 3rd Workshop on Advances in Modal Logics (AIML'2000), Leipzig, Germany, Oct. 2000*. pp. 257–278, World Scientific.

Hladik, J.: 2002, 'Implementation and Optimisation of a Tableau Algorithm for the Guarded Fragment'. In: U. Egly and C. Fermüller (eds.): *Automated Reasoning with Analytic Tableaux and Related Methods*, Vol. 2381 of *Lecture Notes in Computer Science*. pp. 145–159, Springer Verlag.

Hopcroft, J. and J. Ullman: 1979, *Introduction to automata theory, languages, and computation*. Addison-Wesley.

Horrocks, I. and U. Sattler: 1999, 'A description logic with transitive and inverse roles and role hierarchies'. *Journal of Logic and Computation* **9**(3), 385–410.

Horrocks, I. and U. Sattler: 2004, 'Decidability of SHIQ with complex role inclusion axioms'. *Artificial Intelligence* **160**(1–2), 79–104.

Hustadt, U. and R. Schmidt: 2003, 'A Principle for Incorporating Axioms into the First-Order Translation of Modal Formulae.'. In: *CADE'03*, Vol. 2741 of *Lecture Notes in Artificial Intelligence*. pp. 412–426, Springer, Berlin. Long version in (Schmidt and Hustadt, 2004).

Kieronski, E.: 2003, 'The Two-Variable Guarded Fragment with Transitive Guards is 2EXPTIME-hard'. In: A. Gordon (ed.): *6th Int. Conf. on Foundations of Software Science and Computational Structures (FOSSACS'03), Warsaw, Poland*, Vol. 2620 of *Lecture Notes in Computer Science*. pp. 299–312, Springer, Berlin.

Ladner, R.: 1977, 'The computational complexity of provability in systems of modal propositional logic'. *SIAM Journal of Computing* **6**(3), 467–480.

Marx, M., S. Mikulas, and S. Schlobach: 1999, 'Tableau Calculus for Local Cubic Modal Logic and its implementation'. *Logic Journal of the IGPL* **7**(6), 755–778.

Massacci, F.: 1997, 'Tableaux methods for Access Control in Distributed Systems'. In: D. Galmiche (ed.): *TABLEAUX'97*, Vol. 1227 of *Lecture Notes in Artificial Intelligence*. pp. 246–260, Springer, Berlin.

Massacci, F.: 2000, 'Single Steps Tableaux for Modal Logics'. *Journal of Automated Reasoning* **24**(3), 319–364.

Mateescu, A. and A. Salomaa: 1997, 'Formal languages: an introduction and a synopsis'. In: G. Rozenberg and A. Salomaa (eds.): *Handbook of Formal Languages - Volume 1: Word, Language and Grammar*. pp. 1–40, Springer.

Moore, R.: 1977, 'Reasoning about knowledge and action'. In: *IJCAI-5*. pp. 223–227.

Morgan, C.: 1976, 'Methods for automated theorem proving in non classical logics'. *IEEE Transactions on Computers* **25**(8), 852–862.

Nonnengart, A.: 1996, 'Resolution-based calculi for modal and temporal logics'. In: M. McRobbie and J. Slaney (eds.): *13th Conference on Automated Deduction*, Vol. 1104 of *Lecture Notes in Artificial Intelligence*. pp. 599–612, Springer, Berlin.

Ohlbach, H.: 1993, 'Translation methods for non-classical logics: an overview'. *Bulletin of the Interest Group in Propositional and Predicate Logics* **1**(1), 69–90.

Ohlbach, H., A. Nonnengart, M. de Rijke, and D. Gabbay: 2001, 'Encoding two-valued non-classical logics in classical logic'. In: A. Robinson and A. Voronkov (eds.): *Handbook of Automated Reasoning*. pp. 1403–1486, Elsevier Science Publishers B.V.

Orłowska, E.: 1988, 'Relational interpretation of modal logics'. In: H. Andréka, D. Monk, and I. Németi (eds.): *Algebraic logic. Colloquia Mathematica Societatis Janos Bolyai 54*. Amsterdam, pp. 443–471, North Holland.

Perrin, D.: 1990, 'Finite Automata'. In: J. V. Leeuwen (ed.): *Handbook of Theoretical Computer Science, Volume B, Formal models and semantics*. pp. 1–57, Elsevier.

Pin, J.: 1994, 'Logic on words'. *Bulletin of the European Association of Theoretical Computer Science* **54**, 145–165.

Rozenberg, G. and A. Salomaa: 1994, *Cornerstones of Undecidability*, International Series in Computer Science. Prentice Hall.

Sahlqvist, H.: 1975, 'Completeness and correspondence in the first and second order semantics for modal logics'. In: S. Kanger (ed.): *3rd Scandinavian Logic Symposium, Uppsala, Sweden, 1973*. pp. 110–143, North Holland.

Sattler, U.: 1996, 'A concept language extended with different kinds of transitive roles'. In: *20. Deutsche Jahrestagung für Künstliche Intelligenz*. LNM 1137, Springer.

Schmidt, R. and U. Hustadt: 2004, 'A Principle for Incorporating Axioms into First-Order Translation of modal formulae'. Technical Report CSPP-22, The University of Manchester.

Spaan, E.: 1993, 'Complexity of Modal Logics'. Ph.D. thesis, ILLC, Amsterdam University.

Straubing, H.: 1994, *Finite Automata, Formal Logic, and Circuit Complexity*, Progress in Theoretical Computer Science. Birkhäuser.

Szwast, W. and L. Tendera: 2001, 'On the decision problem for the guarded fragment with transitivity'. In: *LICS'01*. pp. 147–156.

Troelstra, A. and H. Schwichtenberg: 1996, *Basic Proof Theory*. Cambridge University Press.

Vakarelov, D.: 1987, 'Abstract characterization of some knowledge representation systems and the logic NIL of nondeterministic information'. In: P. Jorrand and V. Sgurev (eds.): *Artificial Intelligence: Methodology, Systems, Applications*. pp. 255–260, North-Holland, Amsterdam.

van Benthem, J.: 1976, 'Correspondence Theory'. Ph.D. thesis, Mathematical Institute, University of Amsterdam.

van Benthem, J.: 1984, 'Correspondence Theory'. In: D. Gabbay and F. Günthner (eds.): *Handbook of Philosophical Logic, Volume II*. pp. 167–247, Reidel, Dordrecht.

Verma, K.: 2003, 'Automates d'arbres bidirectionnels modulo théories équationnelles'. Ph.D. thesis, ENS de Cachan.

Wolter, F. and M. Zakharyashev: 1997, 'On the relation between intuitionistic and classical modal logics'. *Algebra and Logic* **36**, 121–155.

Deciding the E^+-class by an a posteriori, liftable order

Hans de Nivelle

June 22, 2005

Abstract

We show that the E^+-class can be decided by a liftable order, when it is applied a posteriori. This is a surprising result, because the E^+-class was one of the motivations for the study of non-liftable orders. Also surprising is the behaviour of the resolution process. Initially the variable depth of the clauses may grow, but it will not grow deeper than a certain bound. We do not make use of any type of saturation rule in our completeness proof.

1 Introduction

The E^+ class is the set of clauses, in which all literals have identical variables, and the literals are weakly covering. The class was introduced by [Tam90].
It was shown there (see also [FLTZ93]) that the following order is terminating on E^+ : $A <_v B$ iff the maximal depth at which a variable occurs in A is strictly less than the depth at which a variable occurs in B. Although resolution with this order terminates on the class E^+, it does not satisfy the condition: $A <_v B \Rightarrow A\Theta <_v B\Theta$.
Because of this the completeness of $<_v$ on E^+ was open until [Niv94a], [Niv94b], where a technique for proving completeness of non-liftable orders was introduced, called the resolution game. In fact the E^+ class was one of the motivations for its introduction.
In this paper we show that the E^+ class can be decided by a liftable order after all, when it is applied a posteriori. In [FLTZ93], pp. 109, this is posed as an open problem. (The $R_{<_d}$ refinement occurring there is equal to our $\prec$-ordering)
In [Leitsch97], there is a termination proof for the class E_1, (under the name K), which is essentially E^+ restricted to 1-variable clauses. (The E^+ class is also mentionend there, under the name K^*, on page 223. It is claimed there that saturation techniques are required)
We here prove termination for the full E^+ class, with a fully liftable order, and without saturation rules.

The behaviour of the refinement is very surprising. Initially growth of the depth (and the variable depth) is possible but only up to a certain bound.

1.1 The Resolution Calculus

We briefly introduce the resolution calculus. Contrary to the case of first order logic, we distinguish constants and variables. This makes it possible to omit the $\forall$-quantifiers in clauses.

Definition 1.1 We assume a fixed set of functions/constant symbols F, a fixed set of predicate/propositional symbols P, and a fixed set of variables V. The set of *terms* is recursively defined as follows:

1. A variable is a term.
2. If $t_1, \ldots, t_n$, with $n \geq 0$, are terms, and $f \in F$, then $f(t_1, \ldots, t_n)$ is a term.

If $t_1, \ldots, t_n$, with $n \geq 0$, are terms, and $p \in P$, then $p(t_1, \ldots, t_n)$ is an *atom.*

A *literal* is an atom A, or its negation $\neg\, A$. Atoms of the form A are called *positive.* Atoms of the form $\neg\, A$ are called *negative.*

A *clause* is a finite set of literals.

A term that contains no variables is called *ground.* A term of the form c is called *constant.* A term of the form $f(t_1, \ldots, t_n)$, with $n > 0$, is called *functional.*

Definition 1.2 We define some complexity measures for atoms/clauses/literals:

- Let A be an atom. The *depth* of A is recursively defined as follows:
  - If A is a variable, then $\text{Depth}(A) = 1$.
  - $\text{Depth}(f(t_1, \ldots, t_n))$ equals the maximum of $\{1, 1{+}\text{Depth}(t_1), \ldots, 1{+}\text{Depth}(t_n)\}$.

  The depth of a literal equals the maximal depth of its atoms. The depth of a clause c equals the maximal depth of a literal in c, or 0 for the empty clause.
- Let A be an atom. The *vardepth* of A is recursively defined as follows:
  - If A is ground, then $\text{Vardepth}(A) = -1$.
  - If A is a variable, then $\text{Vardepth}(A) = 0$.
  - Otherwise $\text{Vardepth}(f(t_1, \ldots, t_n)$ equals the maximum of $\{1 + \text{Vardepth}(t_1), \ldots, 1 + \text{Vardepth}(t_n)\}$.

The vardepth of a literal equals the vardepth of its atom. The vardepth of a clause c equals the maximal depth of a literal in c. The vardepth of the empty clause is defined as -1. The vardepth of a set of clauses C is defined as the maximal vardepth of a clause in C.

- Let A be an atom/literal/clause. $\text{Var}(a)$ is defined as the set of variables that occur in A.
- Let c be a clause. $\text{Varnr}(c)$ equals the size of $\text{Var}(c)$. If C is a set of clauses, then $\text{Varnr}(C)$ equals maximal number of variables that occurs in a clause of C.
- Let A be a literal. The *complexity* of A, written as $\#A$ equals the total number of function/constant/variable occurrences in it.

Definition 1.3 A *substitution* is a finite set of variable assignments of the form $\{V_1 := t_1, \ldots, V_n := t_n\}$, such that $V_i \neq t_i$, and $V_i = V_j \Rightarrow t_i = t_j$. The first conditions ensures non-redundancy, the second condition ensures consistency.

We write $A\Theta$ for the effect of Θ on term A.

If Θ_1 and Θ_2 are substitutions, then the *composition* of Θ_1 and Θ_2 is defined as the substitution $\{v := v\Theta_1\Theta_2| \, |v \neq v\Theta_1\Theta_2\}$.

For two literals A and B a unifier is a substitution Θ, such that $A\Theta = B\Theta$. A *most general unifier* Θ is a substitution such that $A\Theta = B\Theta$, and

$$\forall\Theta'(A\Theta' = B\Theta') \Rightarrow \exists\Sigma(\Theta' = \Theta \cdot \Sigma).$$

For the composition holds, for all terms $A(\Theta_1 \cdot \Theta_2) = (A\Theta_1)\Theta_2$.
We present an algorithm for computing the mgu. The algorithm is not efficient because its purpose is proving properties of mgu's, rather than applying it.

Definition 1.4 The following algorithm effectively decides whether or not two literals A and B have a unifier:

Let A and B be two literals, such that $A \neq B$. A *minimal difference* is obtained as follows:

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

The algorithm is defined as follows: Let A and B be the terms to be unified. Put $\Theta = \{\ \}$, the idempotent substitution.

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

Theorem 1.5 The procedure of Definition 1.4 is complete and sound, and strongly normalizing. This means that the outcome does not depend, modulo renaming substitutions, on the minimal difference selected.

Definition 1.6 We define the ordered resolution and factorization rules:
Let $\prec$ be an order on literals. The order is used to select the literals in the clauses that can be used for resolution, or factorization. Only the maximal literals in a clause can be used. There are two moments at which the ordering can be used. The first is before the substitution. This is called *a priori* application. The second is after the substitution, this is called *a posteriori* application.

APRIORI Let $\{A_1\} \cup R_1$ and $\{\neg\ A_2\} \cup R_2$, be two clauses, s.t.

- $\{A_1\} \cup R_1$ and $\{\neg\ A_2\} \cup R_2$ have no variables in common,
- A_1 and A_2 have mgu Θ,
- for no $A \in R_1$, it is the case that $A_1 \prec A$,
- for no $A \in R_2$, it is the case that $A_2 \prec A$.

Then the clause $R_1\Theta \cup R_2\Theta$ is called a *resolvent.*

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

- A_1 and A_2 have an mgu Θ,
- for no $A \in R$ it is the case that $A_1 \prec A$.

The clause $\{A_1\Theta\} \cup R\Theta$ is called a *factor* of $\{A_1, A_2\} \cup R$.

APOSTERIORI Let $\{A_1\} \cup R_1$ and $\{\neg\ A_2\} \cup R_2$, be two clauses, s.t.

- $\{A_1\} \cup R_1$ and $\{\neg\ A_2\} \cup R_2$ have no variables in common,

- A_1 and A_2 have mgu Θ,
- For no $A\Theta \in R_1\Theta$, it is the case that $A_1\Theta \prec A\Theta$,
- For no $A\Theta \in R_2\Theta$, it is the case that $A_2\Theta \prec A\Theta$.

Then the clause $R_1\Theta \cup R_2\Theta$ is called a *resolvent*.
Let $\{A_1, A_2\} \cup R$ be a clause, such that

- A_1 and A_2 have mgu Θ,
- for no $A\Theta \in R\Theta$ is it the case that $A_1\Theta \prec A\Theta$.

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

It is not completely trivial to find an example where a priori and a posteriori application of the order differ, in the case that the order is liftable.

Example 1.7 Let $\prec$ be (a sort of) a Knuth-Bendix order: $A \prec B$ if $\# A < \# B$ and each variable occurring in A occurs at least as often in B.
The following derivations are allowed by a priori application of $\prec$, but they are blocked by a posteriori application:
Resolve $\{p(X, X, Y), q(X, Y, Y)\}$ with $\{\neg\, p(0, 0, s(s(0)))\}$.
Resolve $\{p(X, X, Y, 0), q(X, Y, Y, s(s(s(0))))\}$ with $\{\neg\, p(0, 0, Y, 0)\}$.

Definition 1.8 Let $C = \{c_1, \ldots, c_n\}$ be a clause set. We call C *unsatisfiable* if its first order meaning is unsatisfiable. We call C *propositionally unsatisfiable* if for *every* substitution Θ, the set $C\Theta$ is unsatisfiable.

Theorem 1.9 Let $\prec$ be an order satisfying

$$A \prec B \Rightarrow A\Theta \preceq B\Theta.$$

(the *liftability* condition). Then $\prec$-ordered resolution with factoring is complete.

In fact the order need not be fully liftable. It is sufficient if A and B are restricted to literals that can occur in a derivation, and $A\Theta$ and $B\Theta$ are restricted to literals that can occur in an unsatisfiable instance.
This makes the following more complicated condition:

Definition 1.10 Let $\prec$ be a relation on literals. We call $\prec$ liftable if it satisfies the following conditions:

For every unsatisfiable clause set C, there is a propositionally unsatisfiable set $\overline{C}$ of instances of C, and an order $\prec_g$ on the literals in $\overline{C}$, s.t.

1. for all A and B with $A \prec B$, that can occur together in a clause that is $\prec$-derivable from C, and

2. for all $A\Theta$ and $B\Theta$ that occur in $\overline{C}$, (not necessarily in the same clause)

it is the case that

$$A\Theta \prec_g B\Theta, \text{ or } A\Theta = B\Theta.$$

So $\prec$ need not even be an order by itself, but $\prec_g$ has to be an order.

2 Covering Literals

In order to obtain termination for resolution it is necessary that the literals in the clause sets have a special form: The must be *weakly covering*. Although the definition of this notion is technical, covering literals arise naturally from the Skolemization of decidable classes. We show that covering literals are preserved under resolution, and that the mgu of two covering literals is not larger than the largest of them.

Definition 2.1

- A literal A is *covering* if every functional subterm t of A contains all variables of A.
- A literal A is *weakly covering* if every functional, non-ground subterm t of A contains all variables of A.

Covering and weakly covering literals are typically the result of skolemization, when the prefix ends in an existential quantifier. If an atom $a(\overline{x}, y)$ with in the scope of $\forall \overline{x} y$ is skolemized the result equals $a(\overline{x}, f(\overline{x}))$, which is covering. If $a(\overline{x}, y)$ contains ground terms, then the result is weakly covering.
The main property of (weakly) covering literals is that they do not grow when they are unified. Before we prove this we need a technical property.

Definition 2.2 Let Θ be a substitutution. Θ is a *weak renaming* if for each variable V, the result $V\Theta$ is either ground, or a variable.

So what is excluded are substitutions of the type $\{X := f(X)\}$.

Lemma 2.3 Let A and B be two weakly covering literals, such that $\text{Var}(A) \subseteq \text{Var}(B)$, and $\text{Vardepth}(A) \leq \text{Vardepth}(B)$. Let Θ be the mgu of A and B. Then $\text{Vardepth}(A) = \text{Vardepth}(B)$.

proof:
Suppose that $\text{Vardepth}(A) \neq \text{Vardepth}(B)$. Then $\text{Vardepth}(A) < \text{Vardepth}(B)$. For every substitution Θ, it follows $\text{Vardepth}(A\Theta) < \text{Vardepth}(B\Theta)$, because every variable of A occurs in the deepest term of B. But then A and B cannot

be unified.
end of proof

The following states that when two weakly covering terms are unified, the result is not larger than the largest of them.

Theorem 2.4 Let A and B be weakly covering literals that have an mgu Θ. Let $C = A\Theta = B\Theta$. Then:

1. C is weakly covering,
2. $\text{Vardepth}(C) \leq \text{Vardepth}(A)$, or $\text{Vardepth}(C) \leq \text{Vardepth}(B)$.
3. $\text{Varnr}(C) \leq \text{Varnr}(A)$ or $\text{Varnr}(C) \leq \text{Varnr}(B)$.

proof:
Without loss of generality we consider two cases:

- $\text{Vardepth}(A) = \text{Vardepth}(B)$. Assume that the mgu Θ is constructed by the algorithm of Definition 1.4. We will show that for every variable X, the result $X\Theta$ is ground, or a variable.

  Suppose there were a minimal difference of the form (X, t) (or converse), where t is non-ground, and functional. t has to contain all variables of B, since B is weakly covering. The variable X cannot occur in B, because of the occurs check. Define $\Theta' = \{X := t\}$. We have $\text{Var}(B\Theta') \subseteq \text{Var}(A\Theta')$, and $\text{Vardepth}(B\Theta') < \text{Vardepth}(A\Theta')$. This contradicts Lemma 2.3. By following the iteration of Definition 1.4 it follows that the unifier Θ does not assign a non-ground functional term to any of the variables in A or B.

- $\text{Vardepth}(A) < \text{Vardepth}(B)$. Assume that the mgu Θ is constructed by the algorithm of Definition 1.4.

  Let (t_1, t_2) be a minimal difference based on a term with maximal vardepth in B. (t_1, t_2) must be of the form (X, t), where X does not occur in t, and t is non-ground, functional, and contains all variables of B. Put $\Theta' = \{X := t\}$.

  We have $\text{Var}(B\Theta') \subseteq \text{Var}(A\Theta')$ and $\text{Vardepth}(B\Theta') < \text{Vardepth}(A\Theta')$. By Lemma 2.3 it follows that $\text{Vardepth}(A\Theta') = \text{Vardepth}(B\Theta')$.

end of proof

The following is necessary in order to show that when two clauses resolve, and they consist of weakly covering literals, that then the resolvent consists of weakly covering literals. Let $R_1\Theta \cup R_2\Theta$ be a resolvent of $\{A_1\} \cup R_1\}$ and $\{\neg\, A_2\} \cup R_2$. Theorem 2.4 states that $A_1\Theta$ is weakly covering, but we have proven nothing about the literals in $R_i\Theta$.

Theorem 2.5 Let A and B be literals and let Θ be a substitution such that

1. $\mathrm{Var}(A) \subseteq \mathrm{Var}(B)$,
2. A is weakly covering,
3. $B\Theta$ is weakly covering.

Then $A\Theta$ is weakly covering.

proof:
Let t_2 be a non-ground, functional subterm of $A\Theta$. Let V_2 be a variable of $A\Theta$. We have to show that V_2 occurs in t_2.
There must exist a variable V_1, that occurs in A, and such that $V_1\Theta$ contains V_2. (If V_2 was not introduced by Θ, then $V_1 = V_2$.) There are two possibilities:

1. There is a $t_1[W]$ in A, such that $t_1[W\Theta] = t_2$.
   In that case $t_1[W]$ is non-ground, and functional. because of this $t_2 = t_1[W\Theta]$ contains V_1.
2. There is a W in A, such that t_2 is a subterm of $W\Theta$. (This case includes $t_2 = W\Theta$)
   Because W also occurs in B, the term $B\Theta = t_2$ occurs in $B\Theta$. Because $B\Theta$ is weakly covering, V_2 must occur in t_2.

end of proof
Usually ([FLTZ93]) this theorem is found with a 4th condition: B is weakly covering. We have shown that this condition can be dropped. That this results in a true generalization can be seen by putting: $A = p(X,Y), B = p(X, s(Y)), \Theta = \{Y := s(X,Y)\}$. B is not weakly covering, but $A\Theta$ is.
No more conditions can be dropped from Theorem 2.5, as can be seen from the following examples:

1. $A = p(X,Y),\ B = p(X),\ \Theta = \{X := s(X)\}$.
2. $A = p(X, s(Y)),\ B = p(X,Y),\ \Theta = \{\}$.
3. $A = p(X,Y),\ B = p(X,Y),\ \Theta = \{X := s(X)\}$.

The following states that the side literals can be bound by the literals resolved upon.

Lemma 2.6 If

1. $\mathrm{Var}(A) \subseteq \mathrm{Var}(B)$,
2. A is weakly covering,

3. B is weakly covering,
4. $\text{Vardepth}(A) \leq \text{Vardepth}(B)$,

Then: $\text{Vardepth}(A\Theta) \leq \text{Vardepth}(B\Theta)$.

There is another technical fact that we are going to need, namely that unification does not introduce new ground terms in weakly covering literals, unless the result is completely ground.

Lemma 2.7 Let $C = A\Theta = B\Theta$ be the most general unifier of two weakly covering literals. If C is not ground by itself, then every ground term of C occurs either in A or in B.

proof:
If either A or B is ground, then C will be ground, so let us consider the case where neither A, nor B is ground.
Assume that Θ is obtained by the algorithm of Definition 1.4. Then we can write

$$\Theta = \Sigma_1 \cdot \ldots \cdot \Sigma_n,$$

where each Σ_i is a simple substitution of the form $\{X := t\}$.
We show by induction that if both $A\Sigma_1 \cdots \Sigma_i$ and $B\Sigma_1 \cdots \Sigma_i$ are non-ground, then they do not contain new ground terms. This is clearly true for $i = 0$. Suppose it is true for some i. Then $\Sigma_{i+1} = \{X := t\}$, where t is a term occurring either in $A\Sigma_1 \cdots \Sigma_i$, or $B\Sigma_1 \cdots \Sigma_i$. Hence there is no ground term, not in A or B, in t.
This means that the only manner in which a new ground term can be obtained is when t is ground, and X was the last variable in a functional term. Now if variable X occurs somewhere in a functional term, then this term either contains all variables of $A\Sigma_1 \cdots \Sigma_i$, or of $B\Sigma_1 \cdots \Sigma_i$. (Consequence of the proof of Theorem 2.4) Hence if the last variable is substituted away, either $A\Sigma_1 \cdots \Sigma_i$, or $B\Sigma_1 \cdots \Sigma_i$ is ground. Hence one of $A\Theta$ or $B\Theta$ is ground, and so they are both ground.
end of proof

3 The E^+-Class

We are now ready to introduce the E^+-class.

Definition 3.1 Let C be a clause set. C is in the class E^+ if for every clause $c \in C$ the following holds:

1. For all $A, B \in c$, it is the case that $\text{Var}(A) = \text{Var}(B)$.

2. All literals $A \in c$ are weakly covering.

Sometimes the E^+-class is defined in an extended manner in which the clauses are allowed to consist of more than one component. This is not relevant for our discussion.
It was already known that the E^+-class is decidable, and that it can be decided by the following order, when it is applied a priori:

Definition 3.2 Let A and B be two literals. $A <_v B$ if $\text{Vardepth}(A) < \text{Vardepth}(B)$.

This order is non-liftable, which makes it difficult to prove completeness, but proving termination is fairly easy. With the liftable order $\prec$ that we will introduce, the picture is exactly reversed. Proving termination is difficult, but the completeness is a standard result.
That the order $<_v$ is non-liftable, can be seen from the following: We have

$$p(X, s(0)) <_v p(s(X), 0), \text{ and } p(s(0), X) <_v p(0, s(X)).$$

Applying $\{X := 0\}$ we obtain:

$$p(0, s(0)) <_v p(s(0), 0), \text{ and } p(s(0), 0) <_v p(0, s(0)),$$

which contradicts the fact that $<_v$ is an order.
Showing that $<_v$ cannot satisfy Definition 1.10 is slightly more involved. We need to show that the previous conflict can be enforced by a clause set: Take the clause set

$$C = \{\{p(0, s(X))\}, \{\neg\, p(X, s(0)), p(s(X), 0)\}, \{\neg\, p(s(0), X)\}\}$$

C is unsatisfiable, and has only one unsatisfiable set of ground instances:

$$C_g = \{\{p(0, s(0))\}, \{\neg\, p(0, s(0)), p(s(0), 0)\}, \{\neg\, p(s(0), 0)\}\}$$

Since all the ground instances of the previous example lie in C_g, and there is no other unsatisfiable set of ground instances, the order $<_v$ violates Definition 1.10.

Theorem 3.3 Let C be a set of clauses in E^+. Every clause that can be derived from C, (by unrestricted resolution, or factoring) is also in E^+.

proof:
Follows from Theorem 2.4 and Theorem 2.5.
end of proof

Definition 3.4 We define the following order on literals: $A \prec B$ if

1. Vardepth(A) $<$ Vardepth(B) and Depth(A) $<$ Depth(B).
2. Vardepth(A) $=$ Vardepth(B) $= -1$, and Depth(A) $<$ Depth(B).

Order $\prec$ is not liftable in the strong sense of Theorem 1.9, but it satisfies the conditions of Definition 1.10. This order is called $<_d$ in [FLTZ93], page 106.

Theorem 3.5 Relation $\prec$ is liftable.

proof:
We show that $\prec$ satisfies Definition 1.10. Let C be an unsatisfiable clause set in E^+. The class E^+ is closed under unrestricted resolution with factoring. Because of this we may assume that A and B are weakly covering, and Var(A) $=$ Var(B). Take for $\prec_g$ the following order:

$$A\Theta \prec_g B\Theta \text{ iff } \mathrm{Depth}(A\Theta) < \mathrm{Depth}(B\Theta).$$

Suppose that $A \prec B$. There are two cases:

1. Both A and B are ground. Then $A \prec B$ because of the second case of the definition. The substitution does not affect A and B, and hence $A\Theta \prec_g B\Theta$.
2. Both A and B are non-ground. Then $A \prec B$ because of the first case of the definition. We need to show that Depth($A\Theta$) $<$ Depth($B\Theta$). Let $d =$ Depth($A\Theta$). There are two cases to consider:
   (a) The depth d occurs in a term (which is a constant or variable) created by Θ. Then since Var(A) $=$ Var(B), Vardepth(A) $<$ Vardepth(B), the same term is created by Θ in B, and at a deeper position. Hence Depth($A\Theta$) $<$ Depth($B\Theta$).
   (b) The depth d occurs in a term not created by Θ. Then the term occurred already in A. Because of Depth(A) $<$ Depth(B) there is a deeper term in B, and also in $B\Theta$.

end of proof

The following also holds, although it is irrelevant for completeness:

Lemma 3.6 Relation $\prec$ is an order within a clause.

The termination proof for the $<_v$ order is based on the fact that it is impossible to derive a clause c, with Vardepth(c) $>$ Vardepth(C), or containing more variables than a clause of C. This, together with Lemma 2.7 ensures termination. So it is possible that Depth(c) $>$ Depth(C), for a derived clause, but this is harmless, as this can be caused only by a finite set of ground terms.

Example 3.7 Put $c_1 = \{p(X, Y, s(f(X, Y))), q(X, Y, s(f(X, Y)))\}$, and $c_2 = \{\neg\ p(s(s(0)), X, Y\}$. The resolvent $c = \{p(s(s(0)), Y, s(f(s(s(0)), Y)))\}$ is allowed. We have $\text{Vardepth}(c) \leq \text{Vardepth}(c_1)$, $\text{Varnr}(c) \leq \text{Varnr}(c_1)$, but $\text{Depth}(c) > \text{Depth}(c_1)$, $\text{Depth}(c_2)$. The ground term $s(s(0))$ occurred already in c_2.

This is not true for the a posteriori $\prec$-order, as can be seen from the following example:

Example 3.8 Put $c_1 = \{\neg\ p(X, s(s(s(s(0))))), p(s(X), s(s(s(s(0)))))\}$. Resolving the clause twice with itself is allowed, and results in
$c = \{\neg\ p(X, s(s(s(s(0))))), p(s(s(s(s(X)))), s(s(s(s(0)))))\}$, with $\text{Vardepth}(c) = 4$, and $\text{Vardepth}(c_1) = 1$.

We will show however that it is not possible to continue this initial growth. So this unpleasant behaviour can only occur in clauses with low depth.

Definition 3.9 Let A and B be literals. We define the following notions:

1. $A \sqsubseteq B$ iff
$$\text{Depth}(A) - \text{Vardepth}(A) \leq \text{Depth}(B) - \text{Vardepth}(B).$$
2. $A \equiv B$ iff $A \sqsubseteq B$ and $B \sqsubseteq A$.

We have $p(X, s(0)) \sqsubseteq p(X, s(s(0)))$, but not $p(X, s(0)) \sqsubseteq p(X, 0)$.
We have the following technical fact:

Lemma 3.10 Let A and B weakly covering literals with $\text{Var}(A) = \text{Var}(B)$. Let Θ be a substitution, such that $A\Theta$ is non-ground. Then
$$\text{Vardepth}(A\Theta) - \text{Vardepth}(B\Theta) = \text{Vardepth}(A) - \text{Vardepth}(B).$$

proof:
Because the deepest term in both A and B must contain all variables.
end of proof

Theorem 3.11 The relation $\sqsubseteq$ satisfies the following conditions:

O1 Always $A \sqsubseteq A$.

O2 $A \sqsubseteq B$ and $B \sqsubseteq C$ implies $A \sqsubseteq C$.

L1 For weakly covering A and B, with $\text{Var}(A) = \text{Var}(B)$, for substitutions Θ, such that $A\Theta$ is non-ground,
$$A \sqsubseteq B \Rightarrow A\Theta \sqsubseteq B\Theta.$$

L2 For weakly covering A and B, with $\mathrm{Var}(A) = \mathrm{Var}(B)$, for substitutions Θ, such that $A\Theta$ is non-ground, if $\mathrm{Depth}(A\Theta) > \mathrm{Depth}(A)$, then

$$A\Theta \sqsubseteq B\Theta.$$

proof:

O1 Because

$$\mathrm{Depth}(A) - \mathrm{Vardepth}(A) = \mathrm{Depth}(A) - \mathrm{Vardepth}(A).$$

O2 Evidently

$$\mathrm{Depth}(A) - \mathrm{Vardepth}(A) \leq \mathrm{Depth}(B) - \mathrm{Vardepth}(B),$$

and

$$\mathrm{Depth}(B) - \mathrm{Vardepth}(B) \leq \mathrm{Depth}(C) - \mathrm{Vardepth}(C),$$

implies

$$\mathrm{Depth}(A) - \mathrm{Vardepth}(A) \leq \mathrm{Depth}(C) - \mathrm{Vardepth}(C),$$

L1 Assume that $A \sqsubseteq B$, so

$$\mathrm{Depth}(A) - \mathrm{Vardepth}(A) \leq \mathrm{Depth}(B) - \mathrm{Vardepth}(B).$$

We want to show that

$$\mathrm{Depth}(A\Theta) - \mathrm{Vardepth}(A\Theta) \leq \mathrm{Depth}(B\Theta) - \mathrm{Vardepth}(B\Theta).$$

Write $k = \mathrm{Vardepth}(B) - \mathrm{Vardepth}(A) = \mathrm{Vardepth}(B\Theta) - \mathrm{Vardepth}(A\Theta)$. Note that k is possibly negative. Let $d = \mathrm{Depth}(A\Theta)$. We must show that $\mathrm{Depth}(B\Theta) \geq d + k$. Let T be a deepest variable or constant in $A\Theta$. There are two cases:

- T was introduced by Θ. In that case T was also introduced in $B\Theta$, and T causes there a depth of at least $d + k$.
- T was already present in A. In that case $\mathrm{Depth}(A\Theta) = \mathrm{Depth}(A) = d$, and all terms introduced by the substitution Θ had a depth not larger than $\mathrm{Depth}(A) - \mathrm{Vardepth}(A)$. Because of $A \sqsubseteq B$ it must be the case that $\mathrm{Depth}(B) \geq d + k$.

L2 Write $k = \mathrm{Vardepth}(B) - \mathrm{Vardepth}(A) = \mathrm{Vardepth}(B\Theta) - \mathrm{Vardepth}(A\Theta)$. Let T be a deepest variable or constant in $A\Theta$. Let $d = \mathrm{Depth}(T)$. We have to show that $\mathrm{Depth}(B\Theta) \geq d + k$. Because $\mathrm{Depth}(A\Theta) > \mathrm{Depth}(A)$ this T must have been introduced by the substitution. Because of this T also occurs in $B\Theta$. Then $B\Theta$ must have a depth of at least $d + k$.

end of proof

Property **L1** does not hold when the substitution Θ makes the literals ground. The following example demonstrates this: We have $p(s(X), s(0)) \sqsubseteq p(X, s(0))$, but substitution $\Theta = \{X := s(s(s(0)))\}$ results in $p(s(s(s(s(0)))), s(0)) \not\sqsubseteq p(s(s(s(0))), s(0))$.

Lemma 3.12 Let C be a clause set in E^+. Let k be an integer with $k > \text{Depth}(C)$. For every non-ground clause c that is derivable from C with $\prec$ applied a posteriori, the following holds:

1. If $A, B \in c$, and $\text{Depth}(A),\ \text{Depth}(B) > k$, then $A \equiv B$.
2. If $A, B \in c$, with $\text{Depth}(A) > k$, and $\text{Depth}(B) \leq k$, then $A \sqsubseteq B$.

proof:
We may assume that all derived clauses are in E^+. Initially (1) and (2) are trivially satisfied.
It is sufficient to show that properties (1) and (2) are preserved by substitution, deletion of a literal, and $\prec$-ordered propositional resolution, as factorization and resolution, using $\prec$ a posteriori can be decomposed into these rules.

- Deletion of a literal.
  Let c' be obtained from c by deleting one literal. Whenever $A, B \in c'$ also $A, B \in c$. Because of this A and B have the desired properties.

- Propositional resolution.
  Let c be a $\prec$-ordered, propositional resolvent of the clauses $c_1 = \{A\} \cup R_1$ and $c_2 = \{\neg\ A\} \cup R_2$.

  If both c_1 and c_2 contain no literals with depth $> k$, then also c contains no literals with depth $> k$. Because of this c trivially satisfies (1) and (2).

  If one of c_1, c_2 contains a literal with depth $> k$, then $\text{Depth}(A) > k$. This can be seen from the following:

  Without loss of generality assume that c_1 contains a literal B with $\text{Depth}(B) > k$, but that $\text{Depth}(A) \leq k$. Then evidently $\text{Depth}(A) < \text{Depth}(B)$. Since $B \sqsubseteq A$, by (2), it must be the case that $\text{Vardepth}(A) < \text{Vardepth}(B)$. But then we have $A \prec B$, which is a contradiction, since B would block A.

  Let B_1, B_2 be literals with $\text{Depth}(B_1),\ \text{Depth}(B_2) > k$. If B_1 and B_2 originate together from R_1, or R_2, then (1) and (2) are trivially inherited. So assume, without loss of generality that B_1 originates from R_1 and B_2 originates from R_2. Since $B_1 \equiv A$, and $B_2 \equiv A$, by induction, it must be the case that $B_1 \equiv B_2$.

  Let B, C be literals with $\text{Depth}(B) > k$, and $\text{Depth}(C) \leq k$. Without loss of generality assume that B originates from R_1, and that B_2 orginates from R_2. Then by induction we have $A \equiv B$ and $A \sqsubseteq C$. It follows that $B \sqsubseteq C$.

- Substitution.
  Let Θ be a substitution not making c ground. Let $A\Theta$, $B\Theta$ be a pair of literals in $c\Theta$. Without loss of generality consider the following cases. The other cases are either impossible, or symmetric to one of the cases we consider:

  1. Both $\text{Depth}(A),\ \text{Depth}(B) \leq k$, and both $\text{Depth}(A\Theta),\ \text{Depth}(B\Theta) \leq k$.
     A and B trivially satisfy (1) and (2).
  2. Both $\text{Depth}(A),\ \text{Depth}(B) \leq k$, and both $\text{Depth}(A\Theta),\ \text{Depth}(B\Theta) > k$.
     It follows from Theorem 3.11, L2, that $A\Theta \equiv B\Theta$.
  3. Both $\text{Depth}(A),\ \text{Depth}(B) \leq k$, and $\text{Depth}(A\Theta) > k$, but $\text{Depth}(B\Theta) \leq k$.
     It follows from Theorem 3.11, L2, that $A\Theta \sqsubseteq B\Theta$.
  4. $\text{Depth}(A) > k,\ \ \text{Depth}(B) \leq k$, and $\text{Depth}(A\Theta) > k,\ \text{Depth}(B\Theta) \leq k$.
     By induction $A \sqsubseteq B$. It follows from Theorem 3.11, L1, that $A\Theta \sqsubseteq B\Theta$.
  5. $\text{Depth}(A) > k,\ \text{Depth}(B) \leq k$, and $\text{Depth}(A\Theta),\ \text{Depth}(B\Theta) > k$.
     By induction $A \sqsubseteq B$. It follows from Theorem 3.11, L1, that $A\Theta \sqsubseteq B\Theta$. It follows from L2, that $B\Theta \sqsubseteq A\Theta$. Together this makes $A\Theta \equiv B\Theta$.
  6. $\text{Depth}(A), \text{Depth}(B) > k$, and $\text{Depth}(A\Theta),\ \text{Depth}(B\Theta) > k$.
     It follows from decomposing $\equiv$, twice applying Theorem 3.11, L1, that $A\Theta \equiv B\Theta$.

  It is easy to check that this ensures preservation of (1) and (2).

end of proof

This decomposition works because we use the order a posteriori. When the order $\prec$ is applied apriori, things go wrong:

Example 3.13 Let

$$C = \{p(0, s^3(0))\}, \{\neg\, p(X, s^3(0)), p(s(X), s^3(0))\}.$$

The $\prec$-order allows derivation of each $\{p(s^i(0), s^3(0))\}$ in the case of a priori application.

Now we are ready to prove the main theorem:

Theorem 3.14 Let C be a finite set of clauses in E^+. With a posteriori ordered resolution and factoring based on $\prec$ only a finite set of clauses can be derived from C.

proof:
It is sufficient to show that there exists a bound k on the Vardepth of derivable clauses. We take $k = \text{Depth}(C)$, the depth of a deepest clause in C.
We show that no clause with $\text{Vardepth}(C) > k$, can be introduced. First of all note that even unrestricted factoring cannot increase the variable depth of a clause, since by Theorem 2.4 the subsitution Θ must be a weak renaming on at least one of the literals, and because of this on the whole clause.
It remains to consider the resolution rule. We use the same decomposition as in Theorem 3.12. Let c be the first clause with $\text{Vardepth}(c) > k$. We show that there is no such c. Clause c can be seen as the propositional resolvent of $c_1 = \{A_1\Theta\} \cup R_1\Theta$ and $c_2 = \{\neg\ A_2\Theta\} \cup R_2\Theta$.
Let $B\Theta$ be a literal with $\text{Vardepth}(B\Theta) > k$. We show that $B\Theta$ cannot exist, due to the fact that its presence would block the derivation. First note that $\text{Vardepth}(A_i\Theta) \leq k$, because of Theorem 2.4. Because of this we have $\text{Vardepth}(A_i\Theta) < \text{Vardepth}(B\Theta)$.
Now suppose that $\text{Depth}(A_i\Theta) \geq \text{Depth}(B\Theta) > k$, then it would follow from Theorem 3.12 that $A_i\Theta \sqsubseteq B\Theta$, which is a contradiction with

$$\text{Vardepth}(A_i\Theta) < \text{Vardepth}(B\Theta) \text{ and } \text{Depth}(A_i\Theta) \geq \text{Depth}(B\Theta).$$

So it must be the case that $\text{Depth}(A_i\Theta) < \text{Depth}(B\Theta)$. But then the $\prec$-order would have blocked the derivation.
end of proof

4 Future Research

The obvious question is: Can the decidability results in ([Niv98]) be reproduced by a liftable, a posteriori order?

5 Acknowledgements

Part of the research was carried out during a visit to the Max Planck Institut in Saarbrücken, Germany.

References

[ANB96] H. Andréka, J. van Benthem, I. Németi, Modal Languages and Bounded Fragments of Predicate Logic, ILLC Research Report ML-96-03, 1996.

[BFL94] M. Baaz, C. Fermüller, A. Leitsch, A Non-Elementary Speed Up in Proof Length by Structural Clause Form Transformation, In LICS 94.

[BG90] L. Bachmair, H. Ganzinger, On restrictions of Ordered Paramodulation with Simplification, In CADE 10, pp. 427-441, 1990.

[BGG96] E. Börger, E. Grädel, Y. Gurevich, The Classical Decision Problem, Springer Verlag, Berlin Heidelberg, 1996.

[CL73] C-L. Chang, R. C-T. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1973.

[DG79] B. Dreben, W.D. Goldfarb, The Decision Problem, Solvable Classes of Quantificational Formulas, Addision-Wesley Publishing Company, Inc. 1979.

[FLTZ93] C. Fermüller, A. Leitsch, T. Tammet, N. Zamov, Resolution Methods for the Decision Problem, Lecture Notes in Artificial Intelligence 679, Springer Verlag, 1993.

[Joyn76] W. H. Joyner, Resolution Strategies as Decision Procedures, J. ACM 23, 1 (July 1976), pp. 398-417, 1976.

[Leitsch97] A. Leitsch, The Resolution Calculus, Springer Verlag, 1997.

[McCune95] W. W. McCune, Otter 3.0 Reference Manual and Guide, Argonne National Laboratory, Mathematics and Computer Science Division, can be obtained from **ftp.mcs.anl.gov**, directory **pub/Otter**, 1995.

[Niv94a] H. de Nivelle, Application of Resolution Games to Resolution Decision Procedures, Internal Report 94-50, Department of Mathematics and Computer Science, Delft University of Technology, 1994.

[Niv94b] H. de Nivelle, Resolution Games and Non-Liftable Resolution Orderings, in CSL94, pp. 279-293, Springer Verlag, 1994.

[Niv95] H. de Nivelle, Ordering Refinements of Resolution, Ph. D. Thesis, Delft University of Technology, 1995.

[Niv98] H. de Nivelle, Resolution Decides the Guarded Fragment, Internal Report, to appear, 1998.

[Robins65] J. A. Robinson, A Machine Oriented Logic Based on the Resolution Principle, Journal of the ACM, Vol. 12. pp. 23-41, 1965

[Tam90] T. Tammet, The Resolution Program, Able to Decide some Solvable Classes, in COLOG-88, Springer LNCS, pp. 300-312, 1990.

[Weid96] C. Weidenbach, (Max-Planck-Institut für Informatik), The Spass & Flotter Users Guide, Version 0.55, can be obtained from **ftp.mpi-sb.mpg.de**, directory **pub/SPASS**, 1997.

[Zam72] N.K. Zamov, On a Bound for the Complexity of Terms in the Resolution Method, Trudy Mat. Inst. Steklov 121, pp. 1-10, 1972.

A Resolution-Based Decision Procedure for the Two-Variable Fragment with Equality

Hans de Nivelle [1] and Ian Pratt-Hartmann [2]

[1] Max Planck Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
`nivelle@mpi.mpg-sb.de`
[2] Department of Computer Science,
University of Manchester,
Manchester M13 9PL, UK
`ipratt@cs.man.ac.uk`

Abstract. The two-variable-fragment $\mathcal{L}^2_{\approx}$ of first order logic is the set of formulas that do not contain function symbols, that possibly contain equality, and that contain at most two variables. This paper shows how resolution theorem-proving techniques can be used to provide an algorithm for deciding whether any given formula in $\mathcal{L}^2_{\approx}$ is satisfiable. Previous resolution-based techniques could deal only with the equality-free subset $\mathcal{L}^2$ of the two-variable fragment.

1 Introduction

The two-variable-fragment $\mathcal{L}^2_{\approx}$ is the set of formulas that do not contain function symbols, that possibly contain equality ($\approx$), and that use only two variables. The *two-variable fragment without equality* $\mathcal{L}^2$ is the subset of $\mathcal{L}^2_{\approx}$ not involving the predicate $\approx$. For example, the formula $\forall x \exists y [r(x,y) \wedge \forall x (r(y,x) \rightarrow x \approx y)]$, stating that every element is r-related to some element whose only r-successor is itself, is in $\mathcal{L}^2_{\approx}$ (but not in $\mathcal{L}^2$). Note in particular the 're-use' of the variable x by nested quantifiers in this example. In the same way, it is possible to translate modal formulas into $\mathcal{L}^2$, (without equality) by reusing variables. For example, the modal formula $\Box\Diamond\Box a$ can be translated into $\forall y (r(x,y) \rightarrow \exists x (r(y,x) \wedge \forall y (r(x,y) \rightarrow a(y))))$. No equality is needed for translating modal formulas.

Both two-variable fragments are known to be decidable. That is: an algorithm exists which, given any formula $\phi \in \mathcal{L}^2_{\approx}$, will determine whether ϕ is satisfiable. In [GKV97], decidability of $\mathcal{L}^2_{\approx}$ is proven by analyzing the structure of possible models, and showing that if a formula ϕ has any models at all, then it has a model of size $O(2^{|\phi|})$. This gives in principle a decision procedure in non-deterministic exponential time. However this procedure is probably inefficient in practice. This is caused by inherent problems of backtracking. Any backtracking procedure will be spending time retrying details, that are irrelevant for the truth of the formula. Intelligent backtracking can decrease this problem, but cannot completely remove it. Moreover, a backtracking procedure cannot be prevented from redoing the same work in different branches. Improved implementations might decrease this problem, but it cannot be removed completely.

Opposed to this, a resolution procedure works bottom up, starting with the formula in which one is interested. This means that every clause that is derived is related to the original formula, and hence more likely to be relevant. Additionally, a clause can be seen as a lemma, which can be used many times, and which can be seen as representing the set of its instances. Because of this, the risk of repeated work is decreased.

Another advantage, of a more practical nature, is that resolution decision procedures are close to the standard methods of full first-oder automated theorem-proving, so that existing implementations and optimizations can be used. Indeed only a small modification in a standard prover is needed in order to obtain a resolution decision procedure.

To date, resolution-based decision procedures have been developed for various fragments of first-order logic, including the guarded fragment with equality, the Gödel class, and the two-variable fragment without equality $\mathcal{L}^2$. (See [dN95],[dN00a],[GdN99]).

A resolution theorem prover (for unrestricted first order logic) works as follows: The first order formula is translated into a set of clauses through a clausal normal form-transformation. After that, the resolution prover derives new clauses, using derivation rules. Examples of derivation rules are *resolution, factoring* and *paramodulation.* The prover can also delete *redundant* clauses, using certain deletion rules. Common deletion rules are *subsumption, demodulation* and *tautology elimination.* This process terminates when either the empty clause is derived, or a stable set of clauses is reached. This is a set for which every clause that can be derived, can be immediately deleted by one of the deletion rules. For full first-order logic, there are formulas for which the process will not terminate.

Resolution decision procedures are obtained by first identifying an appropriate clause fragment. Then it is shown that certain restrictions of resolution are complete for the given clause fragment, and that all newly derived clauses are within the clause fragment. After that it is shown that the first order formulas of the fragment under consideration can be translated into the given clause fragment. Finally it is shown that there exists only a finite number of non-redundant clauses in the given fragment.

The strategy that we give in this paper is different from usual decision procedures in the fact that it consists of two stages. First, the clauses are partially saturated under a restricted form of resolution. After that, it is shown that clauses containing equality can be replaced by clauses without equality, without affecting satisfiability. The result is a clause set that corresponds to $\mathcal{L}^2$, the two-variable fragment without equality.

At this point, one could continue the work by using any decision procedure for $\mathcal{L}^2$; we prefer to stay within the resolution framework. we present a novel resolution decision procedure based on a *liftable* order.

The plan of the paper is as follows. Section 2 motivates the search for an efficient decision procedure for $\mathcal{L}^2_{\approx}$. Section 3 shows how equality can be removed from a formula ϕ of $\mathcal{L}^2_{\approx}$ without affecting its satisfiability. Section 4 then presents the new resolution-based algorithm for determining satisfiability of formulas of $\mathcal{L}^2$. We assume familiarity with the standard terminology and the basic techniques of resolution theorem-proving. The reader is referred to [FLTZ93] Ch. 2 for the relevant definitions.

2 Motivation

A logic is said to have the *finite-model property* if any satisfiable formula in that logic is satisfiable in a finite structure. It is easy to see that any fragment of first-order logic having the finite model property is decidable; and indeed, most of the known decidable fragments of first-order logic have the finite model property. (For a comprehensive survey, see Börger, Grädel and Gurevitch [BGG97].) One such fragment of particular interest here is the so-called Gödel class: the set of first-order formulas *without equality* which, when put in prenex form, have quantifier prefixes matching the pattern $\exists^*\forall\forall\exists^*$. Gödel [G33] showed that the Gödel class has the finite model property, and is thus decidable. In the same paper, Gödel claimed that allowing $\approx$ in formulas of the Gödel class would not affect the finite model property, a claim which was later shown to be false by Goldfarb [G84]. Between these two discoveries, Scott [S62] showed that any formula of the two-variable fragment can be transformed into a formula in the Gödel fragment which is equisatisfiable. Relying on Gödel's incorrect claim, Scott concluded decidability for $\mathcal{L}^2_{\approx}$. Of course, what Scott actually showed was the decidability for $\mathcal{L}_2$ only. That the full two-variable fragment does indeed have the finite model property was eventually established by Mortimer [M75].

Most proofs of the finite model property actually yield a bound on the size of a smallest model of a satisfiable formula ϕ in terms of the size of (number of symbols in) ϕ, and the result for the two-variable fragment is a case in point. A satisfiable formula $\phi \in \mathcal{L}^2_{\approx}$ of size n has a model with at most 2^{cn} elements, for some constant c (see Bgg97, pp. 377–381). Therefore, we can determine the satisfiability of ϕ by enumerating all such models, a process which can evidently be completed in nondeterministic exponential time. Moreover, it can be shown using standard techniques that satisfiability in $\mathcal{L}^2_{\approx}$ is in fact NEXPTIME-complete (as, indeed, is satisfiability in $\mathcal{L}^2$). Hence, the complexity of model enumeration agrees with the known worst-case complexity of determining satisfiability in $\mathcal{L}^2_{\approx}$.

Nevertheless, enumeration of models up to a certain size is in practice an inefficient method for determining satisfiability in $\phi \in \mathcal{L}^2_{\approx}$, especially if no model exists. A much more promising method is to adapt a resolution-based theorem prover so that termination on formulas of $\mathcal{L}^2_{\approx}$ is guaranteed. Such an approach has already been employed for other decidable fragments. In particular, the Gödel class can be decided using *ordered resolution*; moreover, as was pointed out in [FLTZ93], by applying Scott's reduction from $\mathcal{L}^2$ to the Gödel class, the same technique can be used to decide $\mathcal{L}^2$ as well. Unfortunately however, this method does not apply to the whole fragment $\mathcal{L}^2_\approx$, since adding equality to the Gödel class leads to undecidability. The main contribution of the present paper is to show that, with the aid of some technical manœuvres, this approach can nevertheless be extended to the whole fragment $\mathcal{L}^2_\approx$. To the best of the authors' knowledge, this is the first really practical decision procedure that has been proposed for the full two-variable fragment.

The fragment $\mathcal{L}^2_\approx$ is of particular interest when dealing with natural language input, because many simple natural language sentences translate into $\mathcal{L}^2_\approx$. To give a somewhat fanciful example, the sentence

Every meta-barber shaves every man who shaves no man who shaves himself

translates to the two-variable formula

$$\begin{aligned}&\forall x(\text{meta-barber}(x) \rightarrow \\ &\quad \forall y((\text{man}(y) \wedge \forall x((\text{man}(x) \wedge \text{shave}(x,x)) \rightarrow \neg\text{shave}(y,x))) \rightarrow \text{shave}(x,y))).\end{aligned}$$

Although by no means all of English translates to the two-variable fragment (just think, for example, of ditransitive verbs), a useful subset of English can nevertheless be treated in this way. In particular, Pratt-Hartmann [PH00] gives a naturally circumscribed fragment of English which is shown to have exactly the expressive power of $\mathcal{L}^2_\approx$. Certainly, the two-variable fragment is more useful when dealing with natural language than other well-known decidable fragments, such as the guarded fragment [AvBN98] or any of the quantifier prefix fragments, whose formulas do not fit translations of natural language constructs easily.

3 Making Equality Disappear

In this section, we give give a method for removing equality from a formula in $\mathcal{L}^2_\approx$, based on resolution. Let ϕ to be some formula of $\mathcal{L}^2_\approx$. We can assume that ϕ contains only unary and binary predicates, since predicates of higher arity—as long as they feature only two variables—can be removed by a transformation (see [GKV97] for details). Throughout this section, if $\psi(x)$ is an $\mathcal{L}^2_\approx$-formula whose only free variable is x, we use the abbreviation $\exists! x\ \psi(x)$ for the $\mathcal{L}^2_\approx$-formula

$$\exists x[\psi(x) \wedge \forall y(\ \psi(y) \rightarrow x \approx y)],$$

asserting that ψ is satisfied by exactly one object.

Occurrences of the $\approx$-symbol fall into two groups. Negative occurrences can be 'simulated' without recourse to equality. Positive occcurrences can be restricted to those belonging to a $\exists!$ quantifier. This is done in the key step of our procedure, which is described in Lemma 4. The remaining occurrences of $\approx$ can axiomatized within $\mathcal{L}^2$. This enables us to remove all occurrences of $\approx$.

Definition 1. *An* atom *is defined as usual. A* literal *is an atom, or its negation. A formula α is in* conjunctive normal form *if it has form $c_1 \wedge \cdots \wedge c_n$, where each $c_i, (1 \leq i \leq n)$ is a disjunction $c_i = \beta_{i,1} \vee \cdots \vee \beta_{i,l_i}$ of literals.*

A formula in CNF is not the same as a set of clauses, because the clauses can be universally quantified. We begin by removing individual constants from our formula ϕ.

Lemma 1. *Let $\phi \in \mathcal{L}^2_{\approx}$, and let the sets of individual constants, unary predicates and binary predicates occurring in ϕ be D, P and R, respectively. Let ϕ' be the result of replacing any atoms in ϕ involving individual constants with formulas according to the following table:*

Atom	*Replacement formula*
$p(d)$	$\exists x(p(x) \wedge p_d(x))$
$r(d, x)$	$\exists y(r(y, x) \wedge p_d(y))$
$r(d, y)$	$\exists x(r(x, y) \wedge p_d(x))$
$r(x, d)$	$\exists y(r(x, y) \wedge p_d(y))$
$r(y, d)$	$\exists x(r(y, x) \wedge p_d(x))$
$r(d, d')$	$\exists x \exists y(r(x, y) \wedge p_d(x) \wedge p_{d'}(y))$

where $p \in P$, $r \in R$, $d \in D$, and where the unary predicates $p_d(x)$ (for $d \in D$) are all new. Then ϕ is equisatisfiable with

$$\phi_0 := \phi' \wedge \textstyle\bigwedge_{d \in D} \exists! x \; p_d(x).$$

In fact, individual constants will be reintroduced later; however removing them at this point makes the key step described in Lemma 4 much easier to follow. Next, we convert to Scott normal form. The following result is standard (see, e.g. [BGG97] lemma 8.1.2).

Lemma 2. *Let ϕ be a formula in $\mathcal{L}^2_{\approx}$. There is an equisatisfiable formula ϕ_1 with form $\phi_1 = \beta_1 \wedge \cdots \wedge \beta_n$, where each β_i has one of the following three forms:*

$$\exists x \; \alpha_i, \qquad \forall x \exists y \; \alpha_i, \qquad \text{or } \forall x \forall y \; \alpha_i.$$

Each α_i is a formula in conjunctive normal form. We call the types of the β_i from left to right Type 1, Type 2, and Type 3. If β_i is a Type 1 formula, then α_i contains at most the variable x. If β_i is a Type 2 or a Type 3 formula, then α_i contains at most the variables x and y. There are no constants in the α_i.

The transformation in question introduces some new predicate letters, but no new individual constants or function symbols. Next, we move all occurrences of $\approx$ out of the α_i into the quantifiers.

Lemma 3. *Let ϕ_1 be as defined in Lemma 2. Formula ϕ_1 can be transformed into a formula $\phi_2 = \beta_1 \wedge \cdots \wedge \beta_n$, where each β_i has one of the following forms:*

$$\exists x \; \alpha_i,$$

$$\forall x \exists y \; (x \not\approx y \wedge \alpha_i),$$

or

$$\forall x \forall y \; (x \approx y \vee \alpha_i).$$

Now each of the α_i is a formula without equality, in conjunctive normal form.

If ϕ_2 contains more than one Type 3 formula, then they can be merged. In the sequel we will assume that this is done, and that there is only one Type 3 formula in ϕ_2.

We come to the key idea of the present transformation—the elimination of the positive occurrence of $\approx$ in a disjunction $\forall x \forall y(\alpha(x,y) \vee \beta(x) \vee \gamma(y) \vee x \approx y)$ of ϕ_2. Our solution is to saturate ϕ_2 partially under resolution, and than to eliminate the disjunctions that have a non-empty $\alpha(x,y)$. When this is done, we have only disjunctions of the form $\forall x \forall y(x \approx y \vee \beta(x) \vee \gamma(y))$. These can be read as: If $\beta(x)$ does not hold everywhere, and $\gamma(y)$ does not hold everywhere, then there is one point c, which is the only point on which $\beta(c)$ does not hold, and also the only point on which $\gamma(c)$ does not hold. This provides the transformation of $\approx$ into $\exists!$.

Definition 2. *We need the following, restricted version of the resolution rule. It is restricted because we allow resolution only between two-variable literals.*

Let $\beta(x,y) \vee r_1$ and $\neg\beta(x,y) \vee r_2$ be disjunctions, occurring in an α_i of ϕ_2. Then $r_1 \vee r_2$ is a resolvent. The r_1 and r_2 are disjunctions of literals. We implicitly assume that $r_1 \vee r_2$ is normalized after the resolution step. That means that multiple occurrences of the same literal are removed.

We allow swapping of variables, but we do not allow proper instantiation. So $\beta(y,x) \vee r_1$ and $\neg\beta(x,y) \vee r_2$ can resolve into $r_1[x \leftrightarrow y] \vee r_2$.

Inside a formula ϕ_2, defined as in Lemma 3, we allow resolution as follows: Resolution is allowed between disjunctions inside the Type 3 formula. The results are added to the Type 3 formula. We also allow resolution between a disjunction inside a Type 2 formula and a disjunction inside a Type 3 formula. The result is added to the Type 2 disjunction that was used.

It is easily checked that the resolution rules are sound. If ϕ' is obtained from ϕ by a resolution step, then $\phi' \leftrightarrow \phi$. Termination follows from the fact that only finitely many normalized disjunctions exist over a given signature.

Lemma 4. *Let ϕ_2 be defined as in Lemma 3. Let ϕ_3 be its closure under resolution, as defined in Definition 2. Let ϕ_4 be obtained from ϕ_3 by removing all disjunctions containing a two-variable literal (other than $\approx$) from the Type 3 formula. Then ϕ_3 has a model iff ϕ_4 has a model*

Proof. It is clear that if ϕ_3 has a model, then ϕ_4 has a model, since resolution is a sound inference rule.

For the other direction, let $\mathfrak{B}$ be a structure in which ϕ_4 is true. We will modify $\mathfrak{B}$ into a new structure $\mathfrak{B}^*$, in which ϕ_3 holds. We use B for the domain of $\mathfrak{B}$. The new structure $\mathfrak{B}^*$ will have the same domain B. It will be obtained from $\mathfrak{B}$ by changing the truth-values of the two-variable predicates. For the rest, $\mathfrak{B}^*$ will be identical to $\mathfrak{B}$.

We assume that both ϕ_3 and ϕ_4 are decomposed as in Lemma 3. We write $\beta_1 \wedge \cdots \wedge \beta_n$ for the decomposition of ϕ_3, and $\beta'_1 \wedge \cdots \wedge \beta'_n$ for the decomposition of ϕ_4. We define α_i and α'_i accordingly. We use t for the index of both Type 3 subformulas. Obviously the β'_i can be arranged in such a way that $(i \neq t) \Rightarrow (\beta_i = \beta'_i)$.

As said before, we intend to reinterprete the two-variable predicates in such a way that β_t becomes true. When doing so, we run the risk of making a β_i of Type 2 false. In order to avoid this we need the following:

For each Type 2 subformula β_i of ϕ_4, we assume a *choice function* f_i of type $B \to B$. It is defined as follows: Because $\beta_i = \forall x \exists y (x \not\approx y \wedge \alpha_i)$ is true in $\mathfrak{B}$, for each $b_1 \in B$, there exists a $b_2 \neq b_1$ in B, s.t. $\mathfrak{B}^* \models \alpha_i(b_1, b_2)$. The choice function f_i is defined by choosing one such b_2 for each b_1.

Let b_1 and b_2 be two distinct elements of B. We define *the pattern* of $\mathfrak{B}$ on $\{b_1, b_2\}$ as the vector of truth values for the binary predicates involving both of b_1 and b_2. So, if $p_1, \ldots, p_r$ are all the binary predicates symbols of ϕ_3, then for each $\{b_1, b_2\}$, the pattern determines whether or not $\mathfrak{B} \models p_j(x,y)(b_1, b_2)$ and whether or not $\mathfrak{B} \models p_j(y,x)(b_1, b_2)$. It does not say anything about the unary predicates, nor about $\mathfrak{B} \models p_j(x,y)(b_1, b_1)$ or $\mathfrak{B} \models p_j(x,y)(b_2, b_2)$.

The intuition of the construction is as follows: If ϕ_3 is not true in $\mathfrak{B}$, this is caused by the fact that there are $(b_1, b_2) \in B$, for which $\mathfrak{B} \not\models \alpha_t(b_1, b_2)$. This must be caused by the fact there is a disjunction $\gamma(x,y) \vee \delta(x) \vee \eta(y) \in \alpha_t$, for which $\mathfrak{B} \not\models \delta(x)(b_1)$, $\mathfrak{B} \not\models \eta(y)(b_2)$, and $\mathfrak{B} \not\models \alpha(x,y)$.
We will change the pattern on $\{b_1, b_2\}$ in such a way that $\alpha(x,y)$ will become true.

Before we can proceed, we need to define the subformulas that are involved: Let $\lambda_1, \ldots, \lambda_i, \ldots, \lambda_p,\ p \geq 0$ be the indices of the Type 2 subformulas of ϕ_3, for which $f_{\lambda_i}(b_1) = b_2$, Similarly, let $\mu_1, \ldots, \mu_j, \ldots, \mu_q,\ q \geq 0$ be the indices of the Type 2 subformulas of ϕ_3, for which $f_{\mu_j}(b_2) = b_1$.

The α_t and the α_{λ_i} and α_{μ_j} are formulas in conjunctive normal form, i.e. conjunctions of disjuctions. We are going to select the disjunctions, whose truth depends on the binary predicates on $\{b_1, b_2\}$. For each $i, (1 \leq i \leq p)$, let $\alpha^1_{\lambda_i}$ be obtained from α_{λ_i} by selecting those disjunctions of which the literals involving one variable are false in $\mathfrak{B}$ on (b_1, b_2). Let $\alpha^2_{\lambda_i}$ be obtained by removing the one-variable predicates from $\alpha^1_{\lambda_i}$. For $j, (1 \leq j \leq q)$, we define $\alpha^1_{\mu_j}$ and $\alpha^2_{\mu_j}$ analogeously.

For α_t we need two copies, because of the two directions involved. Let $\alpha^1_{t_1}$ be obtained from α_t be selecting those disjunctions of which the literals involving one variable are false in $\mathfrak{B}$ on (b_1, b_2). Let $\alpha^2_{t_1}$ be obtained by deleting the one-variable predicates from $\alpha^1_{t_1}$. Similarly, let $\alpha^1_{t_2}$ be obtained by selecting the disjunctions, of which the one-variable predicates are false on (b_2, b_1). Then $\alpha^2_{t_2}$ is obtained by deleting the one-variable predicates from $\alpha^1_{t_2}$.

It is clear that in order to obtain $\mathfrak{B}^*$, it is sufficient to replace in $\mathfrak{B}$ the patterns on each $\{b_1, b_2\}, b_1 \neq b_2$ by a pattern making the $\alpha^2_{\lambda_i},\ \alpha^2_{\mu_j},\ \alpha^2_{t_1},\ \alpha^2_{t_2}$ true on (b_1, b_2). The patterns can be replaced in $\mathfrak{B}$ in parallel.

It remains to show that the patterns exist. This is guaranteed by the fact that α_t and the α_i of Type 2 are sufficiently closed under resolution. Since we are considering a fixed (b_1, b_2), we are dealing with a propositional problem. We apply Lemma 11, using $r = \alpha^2_{t_1} \cup \alpha^2_{t_2}$, and using $c_1 \wedge \cdots \wedge c_m = \alpha^2_{\lambda_i},\ \alpha^2_{\mu_j}$.

It is easily checked that all necessary resolvents are allowed by Definition 2. The $\alpha^2_{\lambda_i}$, $\alpha^2_{\mu_j}$ are consistent, because they are true in $\mathfrak{B}$ on (b_1, b_2). It is also clear $\alpha^2_{t_1} \cup \alpha^2_{t_2}$ cannot contain the empty clause, since it would originate from a disjunction in α_t for which the one-variable literals are false in $\mathfrak{B}$ on (b_1, b_2). But this disjunction would be in α'_t as well, as it does not contain a two-variable literal. This contradicts the assumption that ϕ_4 is true in $\mathfrak{B}$.

Thus, Lemma 4 tells us that, if we saturate ϕ_3 under resolution, then we can delete all the disjunctions $\alpha(x,y) \vee \beta(x) \vee \gamma(y) \vee x \approx y$, for which $\alpha(x,y)$ is non-empty. Although exhaustive application of resolution is computationally expensive, in the context of determining satisfiability in the two-variable fragment, the transformation step from ϕ_3 to ϕ_4 in fact comes for free. This is because existing resolution-based approaches to determining satisfiability in $\mathcal{L}^2$ begin with conversion to Scott normal form, followed by clausification and exhaustive application of ordered resolution *anyway.* All the present procedure requires is that we perform a resolution version of resolution first, and then pause to delete the inseparable clauses, before resuming (pretty well) what we would have done even if no equality were present. It hardly needs mentioning that the elimination of a whole set of clauses requires no computation whatever: in particular, the model-theoretic manipulations used to prove Lemma 4 form no part of the decision procedure for $\mathcal{L}^2_{\approx}$.

The negative occurrences can be easily deleted by introducing a new, non-reflexive predicate neq. We add a formula $\forall x(\neg \text{neq}(x,x))$, and replace each negative equality $x \not\approx y$ by $\text{neq}(x,y)$. We will do this later, we now first concentrate on the positive equalities in the Type 3 formulas.

The remaining steps in our equality-deletion procedure are all straightforward. There is (at most) one positive occurrence of $\approx$ left. In occurs in the Type 3 subformula α_t and the other literals in α_t are unary. The following logical equivalence is simple to verify:

Lemma 5. *Let $\gamma(x)$ be a formula not involving the variable y and let $\delta(y)$ a formula not involving the variable x. Then the formulas*

$$\forall x \forall y(\gamma(x) \vee \delta(y) \vee x \approx y)$$

and

$$\forall x\ \gamma(x) \vee \forall x\ \delta(x) \vee (\exists! x\ \neg\ \gamma(x) \wedge \forall x(\gamma(x) \leftrightarrow \delta(x)))$$

are logically equivalent.

Using this, we can use the splitting rule to decompose the disjunctions of the Type 3 formula. The result is a formula, in all positive occurrences of $\approx$ belong to a $\exists!$ quantifier. These can be eliminated by introducing new individual constants.

Lemma 6. *Let η be a conjunction of constant-free $\mathcal{L}^2$-formulas in prenex form with quantifier prefixes $\forall x$ and $\forall x \exists y$, and for each i ($1 \leq i \leq m$) let ζ_i be a quantifier- and constant-free formula of $\mathcal{L}^2$ not involving the variable y. Define*

$$\begin{aligned}\theta_i :=& \zeta_i(e_i) \wedge \textstyle\bigwedge_p \forall x(\zeta_i(x) \rightarrow (p(x) \leftrightarrow p(e_i)))\wedge \\ & \textstyle\bigwedge_q \forall x \forall y(\zeta_i(x) \rightarrow (q(x,y) \leftrightarrow q(e_i,y)))\wedge \\ & \textstyle\bigwedge_q \forall x \forall y(\zeta_i(x) \rightarrow (q(y,x) \leftrightarrow q(y,e_i)))\end{aligned}$$

where $e_1, \ldots e_m$ are (new) individual constants, p ranges over all unary predicates mentioned in η or any of the ζ_i, and q ranges over all binary predicates mentioned in η or any of the ζ_i. Then the formulas

$$\psi := \eta \wedge \textstyle\bigwedge_{1 \leq i \leq m} \exists! x \zeta_i(x)$$

and

$$\psi' := \eta \wedge \textstyle\bigwedge_{1 \leq i \leq m} \theta_i$$

are equisatisfiable.

Proof. If $\mathfrak{B} \models \psi$, then it is easy to expand $\mathfrak{B}$ to a structure $\mathfrak{B}'$ such that $\mathfrak{B}' \models \psi'$. Conversely, suppose that $\mathfrak{B}' \models \psi'$, where the domain of $\mathfrak{B}'$ is B. Let $b_1, \ldots b_m$ be the denotations of the constants $e_1, \ldots e_m$ in $\mathfrak{B}'$, respectively. Now define a function f on B as follows:

$$f(b) = \begin{cases} b_i \text{ if } \mathfrak{B}' \models \zeta_i[b] \\ b \text{ otherwise.} \end{cases}$$

And define the structure $\mathfrak{B}$ with domain $f(B)$ as follows:

$$\begin{gathered}\mathfrak{B} \models p[f(b)] \text{ iff } \mathfrak{B}' \models p[b] \\ \mathfrak{B} \models q[f(b), f(b')] \text{ iff } \mathfrak{B}' \models q[b, b']\end{gathered}$$

where b ranges over B, p over the unary predicates mentioned in ψ' and q over the binary predicates mentioned in ψ'. Since $\mathfrak{B}' \models \psi'$, $\mathfrak{B}$ is well-defined. It is then easy to see that $\mathfrak{B} \models \psi$.

Theorem 1. *Let ϕ be any formula of $\mathcal{L}^2_{\approx}$. Then the steps described in this section allow us to compute an equisatisfiable formula ϕ^* in the class $\mathcal{L}^2$. Indeed, if ϕ^* is satisfiable over some domain, then ϕ is satisfiable over a subset of that domain. Moreover, ϕ^* can be written as a conjunction of prenex formulas with quantifier prefixes $\exists x$, $\forall x \forall y$ and $\forall x \exists y$.*

Thus, the satisfiability problem for $\mathcal{L}^2_{\approx}$ has been reduced to that for $\mathcal{L}^2$. Moreover, as we have observed, no significant extra computational cost is incurred in this reduction.

Finally, we note that theorem 1 allows us to infer that $\mathcal{L}^2_{\approx}$ has the finite model property from the corresponding fact about the Gödel fragment. This constitutes an alternative to the proofs cited in Section 2.

4 The 2-Variable Fragment without Equality

In this section, we provide a new decision procedure for the two-variable fragment without equality. Practically, the method does not differ from the method given in [dN00a], but the theoretical foundation is different. The method that we give here is based on a liftable order, i.e. an order that is preserved by substitution. The advantage of liftable orders, is that they are better understood theoretically. It is the (far) hope of the authors that this will eventually leed to an understanding of what makes the resolution decision procedures work. At this moment, the termination/completeness proofs are a collection of tricks. One would hope for a real understanding of the relation between model based decision procedures, and resolution based decision procedures. Decision procedures based on liftable orders appear to be a step in this direction.

As said before, procedure makes use of indexed resolution [B71], also called *lock resolution.* It works as follows: one starts with some clause set upon which we want to apply resolution. First, integers are attached to the literals in the initial clause set. This can be done in any arbitrary way; and distinct occurrences of the same literal can be indexed with different integers. After that, the integers can be used by an order restriction for determining which literals can be resolved away. When a resolvent is formed, the literals in the resolvent simply inherit their indices from the parent clauses. In standard versions of lock-resolution, only the literals indexed by a maximal integer can be resolved away. Our procedure is slightly more general: we allow the selection function to look at both the index and the the literal itself. The key property possessed by this selection function is that is is obtained by *lifting* some order $\prec$ on the ground indexed literals, as we will explain below.

4.1 Indexed resolution

Definition 3. *An* indexed literal *is a pair* (L, a)*, where* L *is a literal, and* a *is an element of some index set. We use the index set* $\{0, 1\}$*. We write* $L{:}a$ *instead of* (L, a)*. An* indexed clause *is a finite set of indexed literals.*

The effect of a substitution Θ on an indexed literal $A{:}a$ is defined by $(A{:}a)\Theta = A\Theta{:}a$. The effect of a substitution on a clause is defined memberwise.

In the sequel we use the term *clause* to mean either a clause (in the usual sense of a finite set of literals) or an indexed clause. It will be clear from the context what type of clause we mean. When present, the indices play no role in the semantics of the clause; however, they do play a role in determining which literals are selected. Extending the notion of a selection function to indexed clauses in the obvious way, we define:

Definition 4. *A* selection function Σ *is a function mapping indexed clauses to indexed clauses for which always* $\Sigma(c) \subseteq c$*. For a clause* c*, we call the indexed literals in* $\Sigma(c)$ selected.

Resolution *Let $c_1 = \{A_1{:}a_1\} \cup R_1$ and $c_2 = \{\neg A_2{:}a_2\} \cup R_2$ be clauses, s.t. A_1 and A_2 are unifiable and selected. Let Θ be the most general unifier of A_1 and A_2. Then the clause $R_1\Theta \cup R_2\Theta$ is a resolvent of c_1 and c_2.*

Factoring *Let $c = \{A_1{:}a_1, A_2{:}a_2\} \cup R$ be a clause, s.t. $A_1{:}a_1$ is selected, and A_1, A_2 have most general unifier Θ. Then $\{A_2{:}a_2\Theta\} \cup R\Theta$ is a factor of c.*

In addition to resolution and factoring, our decision procedure for $\mathcal{L}^2$ uses the following rules.

Subsumption If for clauses c_1, c_2 there is a substitution Θ, such that $c_1\Theta \subseteq c_2$, then c_1 *subsumes* c_2. In that case c_2 can be deleted from the database.

Splitting The splitting rule can be applied when a clause c can be partitioned into two non-empty parts that do not have overlapping variables. If c can be partitioned as $R_1 \vee R_2$, then the prover tries to refute R_1 and R_2 independently.

Observe that the subsumption rule has to take the indices into account. The clause $\{p(x,y){:}1\}$ does not subsume the clause $\{p(x,0){:}0\}$.

Definition 5. *Let $\prec$ be an order on ground indexed literals. Let c be a ground clause containing an indexed literal $A{:}a$. An indexed literal $A{:}a$ is* maximal *in c, if there is no literal $B{:}b \in c$ for which $A{:}a \prec B{:}b$.*

We say that a selection function Σ is obtained by lifting $\prec$ *if it meets the condition that, for every clause c and indexed literal $A{:}a \in c$, if c has an instance $c\Theta$ in which $(A{:}a)\Theta$ is maximal, then $A{:}a \in \Sigma(c)$.*

The significance of Definition 5 lies in the following completeness result, which has a simple proof. It can be obtained by modifying the completeness proof for lock resolution of [CL73], or the one in [FLTZ93].

Lemma 7. *Let C be a set of initial clauses. Let Σ be a selection function obtained by lifting some order $\prec$. Let $\overline{C}$ be a set of indexed clauses satisfying the following:*

- *For every clause $c = \{A_1, \ldots, A_p\} \in C$, there exists an indexed clause $d \in \overline{C}$ that subsumes some indexing $\{A_1{:}a_1, \ldots, A_p{:}a_p\}$ of c.*
- *For every clause c that can be derived from clauses in $\overline{C}$, either by resolution or by factoring, there is a clause $d \in \overline{C}$ that subsumes c.*

Then, if C is unsatisfiable, $\overline{C}$ contains the empty clause.

4.2 The S^2-Class

The S^2-class characterizes the format of clauses obtained from clausifications of formulas in the Gödel class. The equality-free formulas, produced by the transformation of Section 3, can be transformed directly into S^2 by Skolemization. Formula in full $\mathcal{L}^2$ can be transformed through the Gödel class, or directly through a structural clause transformation, as is done in [FLTZ93].

Definition 6. *A clause c is in S^2 if it meets the following conditions:*

1. *c contains at most 2 variables, and c contains no nested function symbols.*
2. *If c contains ground literals, then it is a ground clause.*
3. *Each functional term in c contains all variables occurring in c.*
4. *There is a literal in c that contains all variables of c.*

Note that conditions 2 and 4 can be ensured by application of the splitting rule.

The results of section 3 thus suffice to reduce satisfiability in $\mathcal{L}^2_{\approx}$ to satisfiability in S^2. However, closer examination of the transformation allows us to be slightly more specific about the class of clauses we need to consider. Starting with an arbitrary $\mathcal{L}^2_{\approx}$-formula ϕ, theorem 1 guarantees the existence of an equisatisfiable formula ϕ^* which is a conjunction of $\mathcal{L}^2$-formulas in prenex form with quantifier prefixes $\forall x$, $\forall x \forall y$ and $\forall x \exists y$. Skolemizing and clausifying ϕ^* thus yields a collection of clauses in which all function symbols have arity 1. Furthermore, given that the clauses in question are to be indexed in some way, there is nothing to stop us indexing two-variable clauses with index 1 and clauses with fewer than two variables with index 0. This leads us to the definition:

Definition 7. *The class S^{2+i} is defined as the class S^2, but with additional conditions:*

5. *All functions are 1-place.*
6. *If c contains 2 variables, then the indexed literals with 2 variables are exactly the literals with index 1.*

Notice, incidentally, that Conditions 3 and 5 immediately imply that all two-variable clauses are function-free.

At first sight, it might appear that we are making no use of the possibility of giving different indices to distinct occurrences of the same literal. However, resolving $\{p(x,y){:}1, q(x,y){:}1, q(x,x){:}0\}$ with $\{\neg p(x,x){:}1\}$, yields the clause $\{q(x,x){:}1, q(x,x){:}0\}$, in which different indices are used for distinct occurrences of the same literal. To obtain a decision procedure for S^{2+i}, it suffices to define a selection function which is obtained by lifting, and which ensures that all derived clauses are within S^{2+i}. Correctness follows from the soundness of resolution and lemma 7; termination follows from the fact that for a given finite signature, there exist only finitely many non-equivalent clauses in S^{2+i}.

4.3 A decision procedure for the S^{2+i}-Class

We begin by establishing a suitable order on ground literals.

Definition 8. *Let the order $\prec_2$ on ground indexed literals be defined as follows:*

- *If literal A is strictly less deep than B, then $A{:}a \prec_2 B{:}b$.*
- *If literal A and B have the same depth, and $a < b$, then $A{:}a \prec_2 B{:}b$.*

Next, we give the selection function Σ_2 used in the second phase of resolution.

Definition 9. *Every indexed literal $A{:}a$ in a clause c is selected, unless one of the following two conditions holds:*

- *A has no functional terms, $a = 0$, and there is a literal $B{:}b$ in c with $b = 1$.*
- *$A{:}a$ has no functional terms, and there are literals with functional terms in c.*

Lemma 8. *Σ_2 is obtained by lifting $\prec_2$.*

Proof. Let c be an arbitrary clause in S^{2+i}. Let $A{:}a$ be a literal in c that is not selected. We need to show that in every instance $c\Theta$ of c, the literal $A\Theta{:}a$ is non-maximal. First observe that A cannot have any functional terms.
If there is a literal $B{:}b$ with functional terms in c, then this functional term contains all variables of A. So, for every substitution Θ it is the case that $B\Theta$ is deeper than $A\Theta$. As a consequence, $A\Theta{:}a$ is non-maximal in $c\Theta$.
If there is no literal $B{:}b$ with functional terms, then $a = 0$ and there must be a $B{:}b \in c$ with $b = 1$. If c is a one-variable clause, then $A\Theta$ and $B\Theta$ always have the same depth, for every substitution Θ. Because of this $A\Theta{:}0 \prec_2 B\Theta{:}1$. Otherwise Condition 6 applies to c. Literal A contains 1 variable and literal B contains 2 variables. Write $A[X]{:}0$ and $B[X,Y]{:}1$. Let Θ be a substitution. If $Y\Theta$ is deeper than $X\Theta$, then $B[X,Y]\Theta$ is deeper than $A[X]\Theta$ and $A[X]\Theta{:}0 \prec_2 B[X,Y]\Theta{:}1$. Otherwise $A[X]\Theta$ and $B[X,Y]\Theta$ have equal depth and also $A[X]\Theta{:}0 \prec_2 B[X,Y]\Theta{:}1$. This completes the proof.

Note how the indices play an essential role in the definition of Σ_2: no selection function obatined by lifting an ordering on unindexed literals could ensure that $p(x,y)$ is always prefered over $p(x,x)$, since the literals have a common instance.
The next step is to show that resolution with Σ_2 never leads outside S^{2+i}.

Lemma 9. *Each literal selected by Σ_2 contains all variables of its clause, and contains a deepest occurrence of each variable in the clause.*

Proof. If the clause is ground, then the lemma is trivial. If the clause has one variable, then all literals have one variable, by condition 2. Moreover, if there exist any functional literals in the clause at all, any selected clause is functional, and so contains a deepest occurrence of its literal by condition 1. If the clause is two-variable, it must be non-functional by conditions 3 and 5. Thus, any selected literal must have index 1, and hence contains both variables by condition 6.

Lemma 10. *The strategy keeps clauses inside S^{2+i}.*

Proof. It is quite standard to show that every selection function that satisfies the conditions of Lemma 9 preserves Conditions 1-4 of Definition 6. See for example the remark on p. 115 of [FLTZ93]. Condition 5 is obviously preserved by resolution. The only difficulty lies in showing that condition 6 applies to the resolvent of any two clauses in S^{2+i}. Let c by such a resolvent, then, and assume that c is a two-variable clause.

Since all functions are unary, it is obvious that any two-variable literal in c must have come from a two-variable literal in one of the parent clauses, and hence must have index 1.

Conversely, we must show that any 1-indexed literal in c is two-variable. Let the parents of c be c_1 and c_2, let $B_1 \in c_1$ and $B_2 \in c_2$ be the literals resolved upon, and let Θ be the substitution used in this resolution. Without loss of generality, any 1-indexed literal in c may be written $A\Theta{:}1$, where $A{:}1 \in c_1$. By lemma 9, $\mathrm{Vars}(c_1) = \mathrm{Vars}(B_1)$ and $\mathrm{Vars}(c_2) = \mathrm{Vars}(B_2)$. Hence $\mathrm{Vars}(c_1\Theta) = \mathrm{Vars}(B_1\Theta)$ and $\mathrm{Vars}(c_2\Theta) = \mathrm{Vars}(B_2\Theta)$. But we also have $\mathrm{Vars}(B_1\Theta) = \mathrm{Vars}(B_2\Theta)$, whence $\mathrm{Vars}(c_1\Theta) = \mathrm{Vars}(c_2\Theta)$. Thus, $\mathrm{Vars}(c) \subseteq \mathrm{Vars}(c_1\Theta) \cup \mathrm{Vars}(c_2\Theta) = \mathrm{Vars}(c_1\Theta)$. Since c_1 is in S^{2+i}, condition 6 implies $\mathrm{Vars}(A) = \mathrm{Vars}(c_1)$, whence $\mathrm{Vars}(c) \subseteq \mathrm{Vars}(A)$ as required.

Gathering together the lemmas in this section, we have

Theorem 2. *The rules of resolution, factoring and subsumption give us a decision procedure for sets of clauses in the class S^{2+i}.*

This completes the description of the resolution procedure for $\mathcal{L}^2$.
We end the section with a technical lemma that was needed for the proof of Lemma 4.

Lemma 11. *Let $c_1, \ldots, c_m$ and r be sets of propositional clauses. Let r be closed under resolution. Furthermore assume that each possible resolvent between a clause of a c_k and a clause of r is in c_k. Then, if $c_1 \wedge \cdots \wedge c_m$ is consistent and r does not contain the empty clause, then $c_1 \wedge \cdots \wedge c_m \wedge r$ is consistent.*

Proof. The result can be easily obtained from the completeness of semantic resolution. Semantic resolution is obtained by fixing an interpretation I, and by forbidding resolution steps between two clauses, that are both true in I. Semantic resolution is proven complete in [CL73].

Since $c_1 \wedge \cdots \wedge c_m$ is consistent, there is an interpretation I that makes $c_1 \wedge \cdots \wedge c_m$ true. If $c_1 \wedge \cdots \wedge c_m \wedge r$ were inconsistent, then r would contain the empty clause. All clauses that are false in I, must be in r. Hence a resolution step involving a false clause is always allowed.

5 Conclusions

In this paper we have given a practical procedure for deciding satisfiablity in the two-variable fragment with equality. This procedure involves two new contributions. The first is the use of resolution to transform formulas in the two-variable fragment with equality to the two-variable fragment without equality. The second is a new resolution-based procedure for deciding satisfiability in the two-variable fragment without equality based on a selection function obtained by lifting an order on ground indexed literals.

At this moment, handling of constants is unsatisfactory. We hope to be able to adapt Definition 2 and Lemma 4 in such a way that they can handle constants.

References

[AvBN98] Andréka, H., van Benthem, J. Németi, I.: Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, (1998).

[BGG97] Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Springer Verlag, Berlin Heidelberg New York, (1997).

[B71] Boyer, R.S.: Locking: A Restriction of Resolution (Ph. D. Thesis). University of Texas at Austin, (1971).

[CL73] C-L. Chang, R. C-T. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1973.

[FLTZ93] Fermüller, C., Leitsch, A., Tammet, T., Zamov, N.: Resolution Methods for the Decision Problem. LNCS 679, Springer Verlag Berlin Heidelberg New York, (1993).

[GdN99] Ganzinger, H., de Nivelle, H.: A superposition procedure for the guarded fragment with equality. LICS **14**, IEEE Computer Society Press, (1999), 295–303.

[G84] Goldfarb, W.: the unsolvability of the Gödel Class with Identity. Journal of Symbolic Logic, **49**, (1984), 1237–1252.

[G33] Gödel, K.: Collected Works, Volume 1: Publications 1929-1936. Oxford University Press. Edited by Feferman, S., Dawson, J. jr., Kleene, S., Moore, G., Solovay, R., van Heijenoort, J., (1986).

[GKV97] Grädel, E., Kolaitis, Ph. G., Vardi, M.Y.: On the Decision Problem for Two-Variable First-Order Logic. The Bulletin of Symbolic Logic **3-1**, (1997) 53–68.

[HdNS00] Hustadt, U., de Nivelle, H., Schmidt, R.: Resolution-Based Methods for Modal Logics, Journal of the IGPL **8-3**, (2000), 265-292.

[M75] Mortimer, M.: On languages with two variables, Zeitschrift für mathematische Logik und Grundlagen der Mathematik **21**, (1975), 135–140.

[dN95] de Nivelle, H. Ordering Refinements of Resolution, PhD Thesis. Delft University of Technology, (1995).

[dN00] de Nivelle, H.: Deciding the E^{+}-Class by an A Posteriori, Liftable Order. Annals of Pure and Applied Logic **104-(1-3)**, (2000), 219-232.

[dN00a] de Nivelle, H.: An Overview of Resolution Decision Procedures, Formalizing the Dynamics of Information, CSLI **91**, (2000), 115–130.

[PH00] Pratt-Hartmann, I: On the Semantic Complexity of some Fragments of English. University of Manchester, Department of Computer Science Technical Report UMCS–00–5–1, (2000).

[S62] Scott, D.: A Decision Method for Validity of Sentences in two Variables. Journal of Symbolic Logic **27**, (1962), 477.