Resolution for Forward Guarded Fragment

(Rezolucja dla Fragmentu Przedniego Strzeżonego)

Karol Ochman-Milarski

Praca licencjacka

Promotor: prof. Witold Charatonik

Bartosz Bednarczyk

Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Informatyki

5 lutego 2023

Abstract

The Guarded Fragment is a decidable fragment of first-order logic. We are concerned with a further restriction of the Guarded Fragment, called the Forward Guarded Fragment, in which variables appear in atoms only in the order of quantification. The Guarded Fragment can be decided with the resolution method in the double exponential time. We show that the resolution method for the Guarded Fragment can be used to decide Forward Guarded Fragment in exponential time and we provide the implementation.

Fragment Strzeżony to rozstrzygalny fragment logiki pierwszego rzędu. Rozpatrujemy dalsze ograniczenie Fragmentu Strzeżonego, zwane Fragmentem Strzeżonym Przednim, w którym zmiennie w atomach występują jedynie w porządku kwantyfikacji. Fragment strzeżony można rozstrzygać rezolucyjnie w czasie podwójnie wykładniczym. Pokazujemy, że metoda rezolucyjna dla Fragmentu Strzeżonego może zostać zastosowana do rozstrzygania Fragmentu Strzeżonego Przedniego w czasie wykładniczym i przedstawiamy implementację.

Contents

1 Introduction **7**

2 The Forward Guarded Fragment **9**

2.1 Problem definition 10

3 Resolution procedure **11**

3.1 Procedure overview 11

3.2 Clausification 11

3.3 Inference rules 12

3.3.1 Order 12

3.3.2 Normalization 13

3.3.3 Most general unifier 13

3.3.4 Factoring 14

3.3.5 Resolution 14

3.4 Full algorithm 14

4 Completeness **17**

5 Complexity **19**

5.1 Forwardness 19

5.2 Procedure complexity 23

6 Implementation **25**

6.1 User guide 25

Bibliography **27**

Chapter 1

Introduction

Guarded Fragment (GF) was introduced in [1]. It is a fragment of first-order logic, that allows for an unbounded number of quantifiers and variables while remaining decidable. Its satisfiability problem is 2-EXPTIME complete as shown by [2]. In [4] authors introduce a restriction of the Guarded Fragment inspired by the Fluted Fragment [3] called Forward Guarded Fragment (FGF). It restricts the Guarded Fragment with a requirement that variables appear inside atoms in the order of quantification. FGF enjoys EXPTIME complexity for the satisfiability problem and the tree-model property [4].

In [5] authors show how to decide GF with resolution. Here we adapt their work for FGF. We rely on their proof for completeness but derive the new complexity bound. We also provide the implementation.

Chapter 2

The Forward Guarded Fragment

The Forward Guarded Fragment is a restriction of the Guarded Fragment to formulas where variables of atomic formulas are infixes of the series of quantified variables.

Definition 2.1. Let us define the Guarded Fragment (GF) as the smallest subset of first-order logic satisfying:

1. Atomic formulas without function symbols are in GF
2. GF is closed under the use of logical connectives
3. If $\phi(\bar{x}, \bar{y}) \in GF$ where $\bar{x}, \bar{y}$ are all the free variables of ϕ and formula $\alpha(\bar{x}, \bar{y})$ is an atom then also $\exists_{\bar{x}} \alpha(\bar{x}, \bar{y}) \wedge \phi(\bar{x}, \bar{y}) \in GF$ and $\forall_{\bar{x}} \alpha(\bar{x}, \bar{y}) \rightarrow \phi(\bar{x}, \bar{y}) \in GF$

Definition 2.2. To define the Forward Fragment let us first fix a sequence of variables: $x_1, x_2, \ldots$. The Forward Fragment (FF) is then the smallest subset of first-order logic satisfying:

1. Atomic formulas of form $R(x_i, x_{i+1}, \ldots, x_j)$, that is atoms whose variables in order are infixes (without gaps) of the above sequence, are in FF
2. FF is closed under the use of logical connectives
3. If $\phi(x_1, \ldots, x_n) \in FF$ then also $\exists_{x_n} \phi(x_1, \ldots, x_n) \in FF$ and $\forall_{x_n} \phi(x_1, \ldots, x_n) \in FF$

So we use the fixed sequence of variables as the order of quantification. The literals in a formula use infixes of the quantified sequence.

Definition 2.3. The Forward Guarded Fragment (FGF) is the intersection of the Guarded Fragment and the Forward Fragment.

2.1 Problem definition

The goal is to describe a resolution-based procedure deciding the Forward Guarded Fragment. The algorithm should take an FGF sentence as an input and return true or false based on whether the sentence is satisfiable.

Definition 2.4. A first-order logic formula is called *satisfiable* if there exists a model where the formula gets interpreted as true.

Resolution procedure calculates a set of clauses following from the initial formula. An empty clause – a contradiction – gets derived for unsatisfiable formulas. Then a syntactic proof of the negation of the input formula could be derived from the resolution process. The process for arbitrary first-order formulas may not terminate. For the Guarded Fragment though, it has been shown in [5], that the production of new clauses saturates and when it does, the model where the formula holds can be derived from the saturation. For FGF, which is of our interest, the saturation is achieved more quickly. We will need a flavour of resolution called ordered resolution, which restricts which inferences are allowed.

Chapter 3

Resolution procedure

We will use an instance of the procedure from [5]. As FGF is a subset of GF, it can be applied to FGF as well. Stronger restriction on the input formula guarantees faster termination.

3.1 Procedure overview

First, a formula needs to be translated to the CNF form, which is a conjunction of clauses of literals. We represent a CNF formula as a set of clauses. The transformed formula is a starting point for the resolution procedure. Resolution iterates on the set of clauses, inspecting pairs of clauses for possible inferences. There are two ways to make an inference: resolution and factoring, described in the section *Inference Rules*. When an inference is made, a new clause gets added to the set. Thus in the process, the set grows containing increasingly more clauses following from the initial formula. When an empty clause gets derived, it proves the unsatisfiability of the initial sentence. Otherwise, the process stops after no new clauses can be derived. If the resulting clause set contains no empty clauses, the initial formula is satisfiable.

3.2 Clausification

We will describe a sequence of transformations going from formulas in FGF to formulas in CNF. The transformations are standard for first-order logic and preserve satisfiability.

1. First *NNF* is the transformation to the negation normal form (NNF). It works by recursively pushing negation signs toward the atoms. It does that by repeatedly applying rewrites following from the De Morgans' laws. See Section 2.2 from [6] for a description.

2. *Struct*$_\forall$ is the transformation applied to formulas in NNF returning a set of formulas of the form $\forall_{\bar{x}}\phi(\bar{x})$ where ϕ is already without universal quantifiers. The conjunction of formulas from the resulting set is equisatisfiable with the initial formula. It works, by repeatedly replacing subformulas of form $\forall_{\bar{y}}\psi(\bar{x},\bar{y})$ by fresh atoms $a(\bar{x})$ and adding a defining formula $\forall_{\bar{x}}\forall_{\bar{y}}a(\bar{x}) \rightarrow \psi(\bar{x},\bar{y})$. We observe that the sequence $\bar{x}$ followed by $\bar{y}$ is a prefix of the sequence of variables $x_1, x_2, \ldots$.

3. *Skolemization* is the transformation removing all existential quantifiers and replacing the respective quantified variables with fresh function terms. Namely, when a quantifier $\exists_{x_i}$ gets removed, we substitute a new function term $x_i^\alpha(\bar{x}_{1..j})$ for the variable it bound, where α is a unique identifier for the given quantifier and $x_{\bar{1}..j}$ is the sequence of universally quantified variables in the scope. We apply it to every formula in the set resulting from Struct$_\forall$ transformation.

4. Finally *clausification* yields a formula in CNF by treating the formulas under universal quantifiers as propositional logic sentences. We represent it as a set of clauses and make the universal quantification implicit for all the free variables.

For the definitions of the transformations 2, 3 and 4 check the definitions 2.6, 2.7 and 2.8 respectively from [5].

Definition 3.1. Let CNF be a function from the set of FGF formulas to the set of conjunctive sets of clauses obtained by sequencing the above transformations.

3.3 Inference rules

3.3.1 Order

First, we recall the order on literals from [5]. By *Vardepth* of a term we denote the maximal depth at which variable occurs in the term, that is:

1. $\mathit{Vardepth}(A) = -1$ if A is ground

2. $\mathit{Vardepth}(A) = 0$ if A is a variable

3. $\mathit{Vardepth}(f(t_1,\ldots,t_i)) = 1 + \max\{\mathit{Vardepth}(t_1),\ldots,\mathit{Vardepth}(t_i)\}$ if A is a term

By *Vardepth* of a literal $R(t_1,\ldots,t_i)$ we denote the number $1+\max\{\mathit{Vardepth}(t_1),\ldots,\mathit{Vardepth}(t_i)\}$. By *Var* of a literal we denote its set of variables.

Definition 3.2. Let us define the following order $\sqsubset$ on literals.

1. $A \sqsubset B$ if $Vardepth(A) < Vardepth(B)$, or
2. $A \sqsubset B$ if $Var(A) \subseteq Var(B)$.

Even though not an order on the set of arbitrary literals, it is an order among literals from a single guarded clause as taking part in the resolution. For proof see [5].

We also recall the following lemma from [5].

Lemma 3.3. *Every guarded clause c has a $\sqsubset$-maximal literal, and every maximal literal of c contains all variables of c.*

For proof see Lemma 3.7 from [5].

3.3.2 Normalization

We do not want to leave the choice for the most general unifier at the unification step of resolution. For that, we will need the normalizing renaming.

Definition 3.4. We call the following renaming a *normalization* of a literal:

1. Order variable occurrences lexicographically on $(-depth, index)$ where *depth* is the depth at which a position of the variable occurs and *index* is a position from left where the variable occurs when literal is written in standard notation.
2. Greedily assign variables $x_1, x_2, \dots$ in order

It is convenient to define normalization in this way as it is well defined on all first-order logic formulas. In practice, the terms produced in resolution are variables or Skolem terms and the Skolem terms contain every variable of the literal. Therefore, if Skolem terms are present, then the normalization assigns names in order starting from x_1 to the variables of the Skolem terms, which names all variables already. When literal has no Skolem terms, then normalization simply assigns variables in the order of appearance when written. For example these two literals are normalized:

$$R(x_1, x_2), Q(x_3, x_2, f(x_1, x_2, x_3))$$

3.3.3 Most general unifier

Definition 3.5. A *unifier* of two literals/terms is a substitution, that applied to the literals/terms makes them syntactically equal. A *most general unifier* (MGU) is a unifier σ, such that for every unifier τ there is a substitution ϵ so that $\tau = \epsilon \circ \sigma$.

Any most general unifier composed with a renaming substitution is also a most general unifier. We will write $A\theta$ to signify the result of applying substitution θ to literal A and $c\theta$ to signify the result of applying substitution θ to every literal of a clause c.

In the resolution algorithm from [5] we will additionally specify which MGU is used at the unification step, whereas the initial authors left it unspecified. We are allowed to do that as MGUs for a fixed unification problem differ by renamings only, but renamings influence neither the $\sqsubset$-order nor the remaining valid inferences. The lemma below specifies the MGU.

Lemma 3.6. *Let A_1, A_2 be two literals with an MGU θ. Then there exists an MGU θ' such that $A_1\theta' = A_2\theta'$ is normalized.*

Proof. Let θ' be the substitution obtained by applying θ first and then the normalizing renaming of the literal $A_1\theta$. Substitution θ' is an MGU as it is an MGU composed with a renaming. $\square$

We can now describe the rules for inferring new clauses.

3.3.4 Factoring

Definition 3.7. Let $c_1 = \{A_1, A_2\} \cup R$ be a clause, such that A_1 is maximal in c_1 with respect to $\sqsubset$-order from Definition 3.2. and A_1, A_2 have a most general unifier θ such that $A_1\theta = A_2\theta$ is *normalized*. Then the clause $\{A_1\theta\} \cup R\theta$ is called $\sqsubset$-ordered factor of c_1.

3.3.5 Resolution

Definition 3.8. Let $c_1 = \{A_1\} \cup R_1$ and $c_2 = \{\neg A_2\} \cup R_1$ be two clauses, such that both A_1 and $\neg A_2$ are maximal in their respective clauses with respect to the $\sqsubset$-order, ϵ be a variable renaming such that $A_1\epsilon$ does not share variables with A_2, and $A_1\epsilon$ and A_2 have a most general unifier θ such that $A_1\epsilon\theta = A_2\theta$ is *normalized.* Then the clause $R_1\epsilon\theta \cup R_2\theta$ is called an $\sqsubset$-ordered resolvent of c_1 and c_2.

3.4 Full algorithm

The described algorithm is an instance of the resolution procedure from [5].

```
procedure SAT(φ)
    C ← CNF(φ)
    continue ← True
    while continue do
        continue ← False
        for c1, c2 ∈ C × C do
            if c1, c2 resolve into c then
                C ← C ∪ {c}
                continue ← True
            if c1 factors into c then
                C ← C ∪ {c}
                continue ← True
    return {} ∈? C
```

Chapter 4

Completeness

The resolution algorithm derives a set of clauses and answers the satisfiability question based on whether the set contains the empty clause. The algorithm is complete because the empty clause is guaranteed to be derived for unsatisfiable sentences. This is true about the unrestricted resolution and arbitrary first-order logic sentences, but also about the Guarded Fragment and ordered resolution described above as was shown in [5].

Theorem 4.1. *Algorithm SAT decides the satisfiability of FGF sentences.*

Proof. From Theorem 3.20 of [5] we know that SAT decides GF sentences and FGF is a subset of GF. □

Chapter 5

Complexity

The stronger restriction on FGF formulas compared to GF formulas gives a restriction on produced clauses which we call forwardness.

5.1 Forwardness

Definition 5.1. We will write $\bar{x}_{i..j}$ for the gap-free sequence of variables $x_i, x_{i+1}, \ldots, x_j$ and $\bar{x}^{\bar{\alpha}}_{j..k}(\bar{x}_{1..i})$ for the gap-free sequence of Skolem terms

$$x_j^{\alpha_j}(\bar{x}_{1..i}), x_{j+1}^{\alpha_{j+1}}(\bar{x}_{1..i}), \ldots, x_k^{\alpha_k}(\bar{x}_{1..i})$$

.

Definition 5.2. We call a literal *forward* if it is of form

$$(\neg)R(\bar{x}_{i..j}, \bar{x}^{\bar{\alpha}}_{j+1..k}(\bar{x}_{1..j}))$$

for some relation symbol R and a sequence $\bar{x}^{\bar{\alpha}}_{j+1..k} = x_{j+1}^{\alpha_{j+1}}, \ldots, x_k^{\alpha_k}$ of Skolem function symbols assigned in Skolemization to a sequence $\exists_{j+1}, \ldots, \exists_k$ of quantifiers such that the quantifier $\exists_k$ was in scope of quantifiers $\exists_{j+1}, \ldots, \exists_{k-1}$. This includes ground literals. Variable or Skolem term sequences may be empty.

The condition on Skolem symbols says that a sequence $\bar{\alpha}$ is a sequence of identifiers assigned at the Skolemization step to existential quantifiers at some path from the root of the formula to the subformula of the quantifier $\exists_k$.

We will call a clause forward if its literals are forward.

Lemma 5.3. *Let A be a forward literal, ϵ be its normalization and c be a forward clause with only variables from A, such that there is no literal $B \in c$ such that $A \sqsubset B$. Then $c\epsilon$ is forward.*

Proof. Take A, ϵ and c as above.

If c is ground, then $c\epsilon = c$ and therefore $c\epsilon$ is forward. Let us now assume that c is not ground and therefore A is also not ground.

If A contains a Skolem term then it is normalized because the Skolem term contains all variables of A and the variables are ordered starting from 1. In this case $c\epsilon = c$ so $c\epsilon$ is forward.

In the other case the literal A is of form $R(\bar{x}_{i..j})$ for some relation symbol R and indices i, j. Then A after normalization is the literal $R(\bar{x}_{1..j-i+1})$ and normalization is the renaming $x_k \mapsto x_{k-i+1}$ for $k = i, \ldots, j$. Recall that here $1..j-i+1$ denotes the interval from 1 to $j-i+1$. By the assumptions that literal A is no smaller in the $\sqsubset$ order than the literals of the clause c, it has no smaller *Vardepth* than the literals of c and therefore in c there are no Skolem terms. Literals from c are forward, use variables $x_i, \ldots, x_j$ and do not contain Skolem terms. Therefore the arguments of atoms from c are infixes of the tuple $\bar{x}_{i..j}$ and the arguments of atoms from $c\epsilon$ are infixes of the tuple $\bar{x}_{1..j-i+1}$. So $c\epsilon$ is forward. □

The two following lemmas guarantee that only forward clauses get derived in the resolution process.

Lemma 5.4. *If ϕ is an FGF sentence then CNF(ϕ) contains only forward clauses.*

Proof. Take $\phi \in FGF$. Atoms of ϕ are of form $R(x_{i..j})$ for some relation symbol R and indices i, j. Let us consider the steps of CNF. We only need to consider changes to atoms, because the forwardness of a clause is defined by the forwardness of its literals and the forwardness of a literal does not depend on its polarity.

First NNF transforms ϕ into $NNF(\phi)$, which contains the same atoms as ϕ.

Then Struct$_\forall$ introduces new atoms and does not modify existing ones. The transformation replaces subformulas of the form $\forall_{\bar{y}}\psi(\bar{x}, \bar{y})$ by fresh atoms $a(\bar{x})$ while also adding a defining formula $\forall_{\bar{x}}\forall_{\bar{y}}a(\bar{x}) \rightarrow \psi(\bar{x}, \bar{y})$ to the resulting output. The introduced atoms contain variables in the order of quantification, therefore after the transformation, all the resulting formulas have atoms of form $R(x_{i..j})$ for some relation symbol R and indices i, j.

Let us consider Skolemization. We remind that the output of the Struct$_\forall$ transformation is a set of formulas of form $\forall_{\bar{x}_{1..k}}\psi(\bar{x}_{1..k})$ for some indice k, where ψ is without universal quantifiers. We show that the literals in the output of Skolemization are forward. Let A be any literal from any of the Skolemized formulas and B be its atom. Let $B' = R(\bar{x}_{i..j})$ be the corresponding atom before Skolemization. Then the atom B' is under a sequence of quantifiers $\forall_{\bar{x}_{1..k}}, \exists_{x_{k+1}}, \ldots, \exists_{x_l}$ for some $k, l \in \mathbb{N}$. Therefore $B = R(\bar{t})$ for some infix $\bar{t}$ of the sequence of terms $x_1, \ldots, x_k, x_{k+1}(\bar{x}_{1..k}), \ldots, x_l(\bar{x}_{1..k})$. Therefore the literal A is forward.

Lastly, clausification does not modify the atoms. Every atom of $CNF(\phi)$ comes directly from the output of the previous Struct$_\forall$ transformation. Therefore $CNF(\phi)$

contains only forward literals. $\square$

Lemma 5.5. *1. If c_1, c_2 are forward clauses and c is ordered resolvent of c_1 and c_2, then c is forward.*

2. If c_1 is forward clause and c is an ordered factor of c_1, then c is forward.

Proof. We consider resolution first. Let c_1, c_2 be forward clauses and c be their ordered resolvent. Let $A \in c_1$ and $B \in c_2$ be the literals resolved upon. Without the loss of generality let A be the positive literal. Then $A = R(\bar{x}_{k..l}, \bar{x}^{\bar{\alpha}}_{l+1..m}(\bar{x}_{1..l}))$ and $B = \neg R(\bar{x}_{k+s..o}, \bar{x}^{\bar{\alpha'}}_{o+1..m+s}(\bar{x}_{1..o}))$ for some $k, l, m, o \in \mathbb{N}$, $s \in \mathbb{Z}$ and sequences $\bar{\alpha}$, $\bar{\alpha'}$ as in the Definition 5.2. We denoted by k, m the interval of indices appearing in A and by s the shift compared to B. Then l, o mark indices where the sequence of variables turns into a sequence of Skolem terms in A and B respectively. Let us also assume that in A the prefix of variables $\bar{x}_{i..j}$ is no shorter than in B, that is $l - k \geq o - (k + s)$. The two assumptions can be both made without the loss of generality as the polarity of the literals does not impact the unifier. To calculate the most general unifier, let us first rename the variables of A: $x_i \mapsto y_i$ for $i = k, \dots, l$. Call the said renaming τ. The unification problem is:

$$\begin{array}{lllllll} R(& y_k, \dots, & y_{o-s}, & y_{o-s+1}, \dots, & y_l, & x^{\alpha_{l+1}}_{l+1}(\bar{y}_{1..l}), \dots, & x^{\alpha_m}_m(\bar{y}_{1..l})) \\ \doteq & & & & & & \\ \neg R(& x_{k+s}, \dots, & x_o, & x^{\alpha'_{o+1}}_{o+1}(\bar{x}_{1..o}), \dots, & x^{\alpha'_{l+s}}_{l+s}(\bar{x}_{1..o}), & x^{\alpha'_{l+s+1}}_{l+s+1}(\bar{x}_{1..o}), \dots, & x^{\alpha'_{m+s}}_{m+s}(\bar{x}_{1..o})) \end{array}$$

Comparing the terms we get 3 types of equations:

1. $y_i \doteq x_{i+s}$ for $i = k, \dots, o - s$
2. $y_i \doteq x^{\alpha'_{i+s}}_{i+s}(\bar{x}_{1..o})$ for $i = o - s + 1, \dots, l$
3. $x^{\alpha_i}_i(\bar{y}_{1..l}) \doteq x^{\alpha'_{i+s}}_{i+s}(\bar{x}_{1..o})$ for $s = l + 1, \dots, m$

Every MGU of $A\tau$ and B satisfies the equations.

If there are any equations of type 3 and there exists a solution, then necessarily $x^{\alpha_m}_m$ and $x^{\alpha'_{m+s}}_{m+s}$ are the same function symbols, so $s = 0$. Also $\bar{y}_{1..l} = \bar{x}_{1..o}$, so $o = l$ and $y_i = x_i$ for $i = 1, \dots, l$. It follows that the clauses c_1, c_2 are already unified with the identity unification. Literals A and B are also normalized because Skolem terms contain all variables and the variables are ordered starting from 1. Therefore every literal in the clause c comes directly from one of c_1, c_2, so c is forward.

Let us consider the other case. Then there are no equations of type 3 and A does not contain function terms. In this case, the unification has an easy solution. We will define the unifying substitution $\sigma : \{y_k, \dots, y_l, x_1, \dots, x_o\} \to \{x_1, \dots, x_o, x^{\alpha'_{o+1}}_{o+1}(\bar{x}_{1..o}), \dots, x^{\alpha'_{m+s}}_{m+s}(\bar{x}_{1..o})\}$ Let σ be the identity substitution on variables x_1 to x_o. Therefore $B\sigma = B$. Equations of type 1 and 2 define the substitution on variables y_k to y_l:

- $y_i \leftarrow x_{i+s}$ for $i = k, \ldots, o - s$
- $y_i \leftarrow x_{i+s}^{\alpha'_{i+s}}(\bar{x}_{1..o})$ for $i = o - s + 1, \ldots, l$

The substitution σ is trivially a unifier of $A\tau$ and B, as σ unifies all pairs of relation symbol arguments at matching indices. The unifier σ is the most general unifier, because every unifier more general has to assign a variable instead of a function term to one of the variables y_{o-s+1} to y_l, thus invalidating the corresponding equation. The unifier σ is though not necessarily the one used in the resolution to derive c, as it does not necessarily normalize the literals $A\tau$ and B. Let ϵ be the normalization of the literal $B = B\sigma$, meaning that $\epsilon \circ \sigma$ is the MGU used to resolve c.

Let us now consider the clause $c' = (c_1 \setminus \{A\})\tau\sigma \cup (c_2 \setminus \{B\})\sigma$, that is a clause such that $c = c'\epsilon$. Let us first note that, as the literal B is maximal in c_2 and therefore contains all the variables of c_2, the substitution σ is the identity on c_2. Therefore $(c_2 \setminus \{B\})\sigma$ is forward as a subset of c_2 which is forward. Furthermore, the literal A is a maximal literal in c_1, so its renaming $A\tau = R(\bar{y}_{k..l})$ is a maximal literal in $c_1\tau$. Therefore literals in $c_1\tau$ contain only variables $y_k, \ldots, y_l$ and do not contain non-ground Skolem terms. Also, the literals of c_1 are forward. It follows that every literal in $c_1\tau$ is either ground or it is of the form $(\neg)Q(\bar{y}_{i..j})$ for some infix $\bar{y}_{i..j}$ of $\bar{y}_{k..l}$ and some relation symbol Q. After substitution σ every literal is either ground or it is of the form $(\neg)Q(\bar{t})$ for some infix $\bar{t}$ of the sequence of terms $\bar{x}_{k+s..o}, \bar{x}_{o+1..m+s}^{\bar{\alpha'}}(\bar{x}_{1..o})$, therefore is forward. We conclude that $(c_1 \setminus \{A\})\tau\sigma$ is forward and in turn c' is forward.

To show that c is forward, we will use Lemma 5.3. We know that c' is forward, ϵ is the normalization of B, $c = c'\epsilon$ and B contains all the variables of c'. What remains to be shown is that B is no smaller in the $\sqsubset$ order than the literals in c'. The literal $A\tau$ does not contain function terms and is maximal in $c_1\tau$, so the literals in $c_1\tau$ do not contain non-ground function terms. The substitution σ substitutes for variables of $c_1\tau$ arguments of the atom of B. Therefore *Vardepth* of the literals in $c_1\tau\sigma$ is no greater than the *Vardepth* of B. Also, B contains all the variables of $c_1\tau\sigma$, so it is no smaller than literals in $c_1\tau\sigma$. The literal B is also maximal in $c_2\sigma = c_2$, so we conclude that it is no smaller than literals in the clause c'. From Lemma 5.3 it follows that c is forward.

Let us now consider factoring. Let c_1 be a forward clause and c be its factor. Let A_1, A_2 be the literals participating in the factoring and A_1 be the maximal one. Literal A_1 contains all the variables of A_2 by Lemma 3.3 and both literals are forward. It follows that the literals are identical or A_2 is ground. If A_2 is ground then c is also ground and therefore c is forward. Otherwise, let ϵ be the normalization of A_1. Then $c = (c_1 \setminus \{A_2\})\epsilon$. From Lemma 5.3 applied to the literal A_1, its renaming ϵ and the forward clause $c_1 \setminus \{A_2\}$ we have that c is forward. $\square$

Lemma 5.6. *Let ϕ be an FGF sentence with l existential quantifiers and A be the set of Skolem function symbols in $CNF(\phi)$. Let m be the number of relation symbols* *in ϕ, a be the maximal arity of relation symbols and n be the number of variables in ϕ. Then there are at most $2 \cdot m \cdot n^2 \cdot l$ forward literals using relations from ϕ and function symbols from A.*

Proof. A forward literal $(\neg)R(\bar{x}_{i..j}, \bar{x}^{\bar{\alpha}}_{j+1..k}(\bar{x}_{1..j}))$ begins with a possibly negated relation symbol giving $2 \cdot m$ options. Then one of n^2 infixes of the sequence $x_1, \ldots, x_n$ follows. Then a sequence of Skolem terms follows. The sequence ends with some Skolem term $x_k^{\alpha_k}(\bar{x}_{1..j})$ and that term already defines the sequence of Skolem terms $\bar{x}^{\bar{\alpha}_{j+1..k-1}}_{j+1..k-1}(\bar{x}_{1..j}))$, because in ϕ the quantifier identified by α_k is in the scope of exactly one sequence of quantifiers identified by some $\alpha_{j+1}, \ldots, \alpha_{k-1}$. Therefore there are at most $2 \cdot m \cdot n^2 \cdot l$ forward literals from the lemma. □

5.2 Procedure complexity

Theorem 5.7. *Procedure SAT works in exponential time with respect to the length of the input formula.*

Proof. The complexity is made up of the complexity of the CNF transformation plus the complexity of the following resolution process. Let n be the length of the input formula. Let us consider clausification first:

1. NNF works in linear time with respect to the length of the formula and increases the size of the formula by a constant factor.
2. The output of the $\text{Struct}_\forall$ transformation is a set of formulas. There are at most as many formulas as there are improper subformulas of the input and they are no bigger than the input, so the output is at most of the quadratic size compared to the input. Therefore $\text{Struct}_\forall$ works in at most the quadratic time.
3. The Skolemization works in the linear time and increases the formula by a constant factor.
4. Finally, clausification may produce exponentially many polynomially sized clauses.

Clearly, the complexity of CNF translation is no bigger than exponential. Then the resolution process starts. The process ends once all possible clauses get derived. From Lemma 5.4 and Lemma 5.5 we know that all the derived terms will be forward. From Lemma 5.6, we know that there are at most $2 \cdot n^4$ forward terms. Therefore there are at most $k = 2^{2 \cdot n^4}$ forward clauses. Algorithm SAT has to inspect every possible pair of clauses to derive a new clause or terminate. A forward literal is at most quadratically sized wrt. to n as the arity of both the relation symbols and function symbols is bounded by n. Inspecting a pair of clauses takes polynomial time with respect to n, because both the cardinality of the clause and the sizes of the literals are polynomial. Therefore a new clause gets derived after k^2 polynomially sized steps and the algorithm terminates after producing no more than k clauses. We conclude that SAT works in the exponential time with respect to n. $\square$

Chapter 6

Implementation

6.1 User guide

The implementation is a library written in the Scala programming language. It is compiled with the standard *sbt* tool as described in the projects `README.md`. It provides a data type `GFFormula` common for both the GF formulas and the FGF formulas. Users may want to use the smart constructor `fgfSentence` to construct a `GFFormula` value that is a valid FGF sentence. The library also provides a function `SAT` of type $\mathit{GFFormula} \rightarrow \mathit{Bool}$ implementing the algorithm SAT 3.4.

Bibliography

[1] H. Andréka, I. Németi, J. van Benthem, 1998. Modal Languages and Bounded Fragments of Predicate Logic. Journal of Philosophical Logic, 27(3), 217–274

[2] E. Grädel, 1999, On the Restraining Power of Guards. The Journal of Symbolic Logic, 64(4), 1719–1742, https://doi.org/10.2307/2586808

[3] I. Pratt-Hartmann , W. Szwast, L. Tendera, 2019, The fluted fragment revisited. The Journal of Symbolic Logic

[4] B. Bednarczyk, 2021, Exploiting Forwardness: Satisfiability and Query-Entailment in Forward Guarded Fragment, Logics in Artificial Intelligence 17th European Conference, JELIA 2021, https://dx.doi.org/10.1007/978-3-030-75775-5

[5] H. de Nivelle, M. de Rijke, 2003, Deciding the guarded fragments by resolution. Journal of Symbolic Computation, 35(1), 21-58, https://doi.org/10.1016/S0747-7171(02)00092-5

[6] A. Leitsch, 1997, The resolution calculus. Texts in Theoretical Computer Science, Springer, https://doi.org/10.1007/978-3-642-60605-2