The present work was submitted to the LuFG Theory of Hybrid Systems

BACHELOR OF SCIENCE THESIS

QUANTIFIER ELIMINATION USING THE VIRTUAL SUBSTITUTION

Boris Schüpp

Examiners:
Prof. Dr. Erika Ábrahám
Prof. Dr. Jürgen Giesl
Additional Advisor:
Jasper Nalbach

Aachen, 30 September 2021

Abstract

Quantifier elimination is an interesting topic with various real life applications such as applications by Siemens (quantifier elimination for parameter synthesis in testing purposes) or problem simplification in Geogebra (mathematics/geometry software for students from primary school to university level).

In this thesis a new approach for quantifier elimination and parameter synthesis especially for non-linear real arithmetic has been explored. Using the existing SMT solvers implemented in SMT-RAT, a sample is generated to which then a sample-based virtual substitution approach is applied which may significantly reduce the complexity of standard virtual substitution. Therefore, an equivalent quantifier free formula is generated, which shows the boundaries in regard to the non-quantified variables. The correctness of the method has been documented and the implementation has been verified on several examples.

Acknowledgements

I am very grateful for the opportunity to work on this exciting topic. Special thanks go to professor Ábrahám and Jasper Nalbach for the extensive discussions and support during this project. I also thank professor Giesl for being the second examiner.

Contents

Chapter 1

Introduction

Quantifier elimination is an interesting topic since it yields major simplification to a given formula and can be applied in various contexts. Instances in which the problem can be expressed as a quantified formula appear in various disciplines ranging from safety applications to chemistry and life sciences.

For this particular work, applications by Geogebra, a mathematics software for students from primary school to university level in which the simplification of example problems can be done via quantifier elimination and by Siemens, where the synthesis of appropriate test parameters can be achieved by quantifier elimination, have been discussed.

Whether or not a quantified formula can be efficiently and successfully converted into an equivalent formula that does not require and/or contain any quantifiers depends on the logic that is considered. The possibility of quantifier elimination within a logic is also deeply connected to the problem of decidability of that logic. For nonlinear real arithmetic (NRA), which will be the logic focused on in this thesis, it has been shown that quantifier elimination is possible by Tarski [Tar49]. Starting from that proof, several methods of effective quantifier elimination have been implemented over the past decades. The most prominent approaches are the cylindrical algebraic decomposition [Col75] and the virtual substitution [Wei97]. The principles behind the virtual substitution will be explained in Chapter 2. The virtual substitution has been incorporated into satisfiability modulo theories solving software such as SMT-RAT [Cor16]. A more detailed history of different techniques and improvements of those can be found in [Stu17].

The resulting quantifier-free formula in case of virtual substitution often has a greater complexity than the original formula or other equivalent descriptions of the desired result. A possible solution for this problem is presented in this thesis: In cooperation with an SMT solver, a sample-based virtual substitution approach has been implemented that reduces the virtual substitution to a certain subset of cases for which the SMT solver has previously produced a sample.

Chapter 2

Preliminaries

2.1 Real arithmetic

Let $\mathbb{R}$ be the set of the real numbers, $\mathbb{N}$ the set of natural numbers including zero and let x be a variable that can be assigned with any rational number.

Definition 2.1.1. *A polynomial p is an expression that can be constructed inductively as follows:*

$$p := x \mid c \mid p' + p'' \mid p' - p'' \mid p' \star p''$$

where $c \in \mathbb{R}$ and p' and p'' themselves are polynomials.

We denote the set of variables appearing in p as $\mathrm{Var}(p)$. A polynomial p will be called univariate in case it only contains one variable, therefore $|\mathrm{Var}(p)| = 1$ holds. In case $|\mathrm{Var}(p)| > 1$ the polynomial p will be denoted as multivariate. The set of all univariate polynomials with $\mathrm{Var}(p) = x$ and coefficients from $\mathbb{R}$ is written as $\mathbb{R}[x]$. Analogously, the set of multivariate polynomials that contains variables from the set $\{x_0, \ldots, x_n\}$ is called $\mathbb{R}[x_0, \cdots, x_n]$.

Theorem 2.1.1. *Every polynomial p, with $Var(p) = \{x_0, \ldots, x_n\}$ can be written in a normal form as follows:*

$$p = \sum_{i=0}^{m} c_i \prod_{x_j \in Var(p)} x_j^{e_{ij}}$$

where $c_i \in \mathbb{R}$ (with $0 \leq i \leq m$) and $e_{i,j} \in \mathbb{N}$ (with $0 \leq i \leq m$ and $0 \leq j \leq n$).

Definition 2.1.2. *The degree of a variable x_k in the polynomial p will be defined as:*

$$deg(p,x_k) := \begin{cases} 0, & \text{if } x_k \notin Var(p) \\ max(e_{i,k} | i \in \{0, \ldots, m\}), & \text{if } x_k \in Var(p) \end{cases}$$

If for all x_k holds that $\deg(p,x_k) \leq 1$, we call p linear. Otherwise we call p non-linear. In the case that $\deg(p,x_k) \leq 2$ holds for all x_k and that there is at least one $x_k \in \mathrm{Var}(p)$ such that $\deg(p,x_k) = 2$, p will be called quadratic (in x_k).

Definition 2.1.3. *A constraint cons is a relation defined as follows:*

$$cons := p_1 \sim p_2$$

where p_1 and p_2 are polynomials and $\sim \in \{\leq, <, =, \neq, >, \geq\}$.

Any constraint $p_1 \sim p_2$ can always be rewritten as a constraint $p_3 \sim 0$, with p_1, p_2, p_3 being polynomials and $\sim \in \{\leq, <, = , \neq, >, \geq\}$.

Definition 2.1.4. *A formula φ from real arithmetic can be constructed inductively as follows:*

$$\varphi := cons \mid \varphi' \wedge \varphi'' \mid \neg\varphi' \mid \exists x \varphi'$$

where φ' and φ'' themselves are quantifier-free formulas from real arithmetic, cons is a constraint of the type $p \sim 0$ and x is a variable.

It is important to note that since $\{\neg, \wedge\}$ is functionally complete, every other formula that might contain operators such as $\vee$ or $\rightarrow$ can be described in an equivalent way using the given definition. Additionally, only the existential quantifier is given here since the universal quantifier can be expressed in an equivalent way using a double negation.

Let constraints(φ) be the set of constraints appearing in φ and Polynomials(φ) the set of polynomials found in a formula φ. We extend the definition of Var(p) for formulas as $\mathrm{Var}(\varphi) := \bigcup_{p \in \mathrm{Polynomials}(\varphi)} \mathrm{Var}(p)$. The set of non-quantified variables in φ will be denoted as free(φ) or sometimes *parameters*, while the set of quantified variables is called quant(φ).

Definition 2.1.5. *An assignment α is a function defined as follows:*

$$\alpha : \{x_0, \ldots, x_n\} \rightarrow \mathbb{R}$$

where $\{x_0, \ldots, x_n\}$ denotes a set of variables.

We call an assignment α *matching* in regards to a formula φ if $\mathrm{dom}(\alpha) \supseteq \mathrm{Var}(\varphi)$. We call the set of all possible matching assignments Asm(φ) for a formula φ and Asm(p) for a polynomial p respectively. The *interpretation* of a formula φ under a (matching) assignment will be displayed as $[\![\varphi]\!]^\alpha$; similarly the *evaluation* of a polynomial under an assignment is shown as $[\![p]\!]^\alpha$. Note the difference between interpretation and evaluation: The interpretation always yields a formula equivalent to *true* or *false*, while the evaluation yields a real number. The interpretation effectively replaces every appearance of a variable by the real value given by the assignment. The appearing operators are evaluated in the usual way. In case $[\![\varphi]\!]^\alpha \equiv true$ holds we say that α *satisfies* φ or in short $\alpha \vDash \varphi$. In the latter case we may call a satisfying assignment a (satisfying) *sample*. If a $\alpha \in \mathrm{Asm}(\varphi)$ for a formula φ exists so that $\alpha \vDash \varphi$, then φ is called *satisfiable* (SAT), otherwise it is called *unsatisfiable* (UNSAT).

2.2 Virtual Substitution

Virtual substitution is a quantifier elimination procedure that has been proposed by Weispfenning [Wei97]. However, it has an important restriction: The degree of the variable to be eliminated should not exceed two in any of the given constraints.

2.2.1 Univariate case

To explain the virtual substitution, this problem will be motivated starting from an univariate polynomial. The number of zeros of a univariate polynomial is finite and bound by its degree. Since a polynomial is of continuous nature, a change of sign can only happen at one of these zeros.

Definition 2.2.1. *The sign-function* $sgn : \mathbb{R} \to \{-1,0,1\}$ *is defined as:*

$$sgn(x) := \begin{cases} 1, & if\ x > 0 \\ 0, & if\ x = 0 \\ -1, & if\ x < 0 \end{cases}$$

Definition 2.2.2. *A region* R *is a connected, non-empty subset of* $\mathbb{R}^n$ *with* $n \in \mathbb{N}\backslash\{0\}$.

Note here that the usual condition of a region to be open has been omitted to allow a single value, such as a zero of a polynomial, to be seen as a region.

Definition 2.2.3. *An assignment* α *lies within a region* R *if the number of variables in* α *matches* n *from the definition of* R *and (under assumed ordering of the variables)* $(\alpha(x_0), \dots \alpha(x_n)) \in R$ *holds.*

Definition 2.2.4. *A sign-invariant solution interval* $Sol(p,R)$ *with underlying region* R *for a polynomial* p *is a set of assignments with* $Sol(p) \subseteq Asm(p)$ *such that any* $\alpha \in Sol(p)$ *lies within* R *and* $sgn([\![p]\!]^{\alpha_1}) = sgn([\![p]\!]^{\alpha_2})$ *is given for any* $\alpha_1, \alpha_2 \in Sol(p)$.

Using the zeros of a polynomial, the set of all assignments $\mathrm{Asm}(p)$ can be divided into sign-invariant solution intervals that consist of the zeros themselves, the intervals between the zeros and the intervals to the left of the left-most zero and to the right of the right-most zero. This is shown for an univariate case in Example 2.2.1.

Example 2.2.1. *Consider the univariate polynomial* $p := (x-1) \cdot (x+1)$. *From the zeros* $x_1 = +1$ *and* $x_2 = -1$ *the underlying regions for sign-invariant solution intervals are as follows:*

$$\begin{aligned} R_0 &= (-\infty, -1) \\ R_1 &= [-1, -1] \\ R_2 &= (-1, +1) \\ R_3 &= [+1, +1] \\ R_4 &= (+1, +\infty) \end{aligned}$$

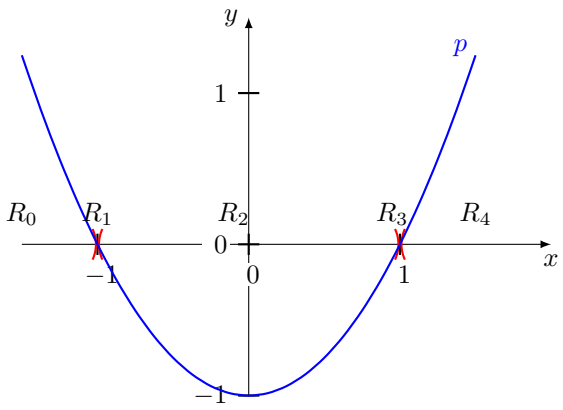

Figure 2.1: Graphical description of sign-invariant solution intervals of p.

Based on this and given a constraint $p \sim 0$ follows that the consideration from one assignment out of each sign-invariant solution interval is sufficient to conclude satisfiability of a formula. For the univariate case this is straightforward since an ordering of the zeros as well as the picking of suitable assignments within the solution intervals can be done easily.

2.2.2 Multivariate case

Treating a multivariate polynomial as a polynomial of the variable that should be eliminated, x, with polynomial coefficients and restricting the degree to at most two yields the following description:

$$p := p_a \cdot x^2 + p_b \cdot x + p_c$$

Here p_a, p_b and p_c are possible multivariate polynomials that do not contain x. The zeros in respect of such a polynomial are *symbolic* and contain a square-root expression as well as a side condition.

Definition 2.2.5. *A square-root expression sqrtEx is described by a term such that*

$$sqrtEx = \frac{q + r \cdot \sqrt{t}}{s}$$

with q, r, t and s being polynomials.

Definition 2.2.6. *The zeros of $p := p_a \cdot x^2 + p_b \cdot x + p_c$, denoted by $\{\xi_0,\xi_1,\xi_2\}$ and their respective side conditions $(sc(\xi_i))$ are*

Zero	Square-root expression	Side condition
ξ_0	$-\frac{p_c}{p_b}$	$p_b \neq 0 \wedge p_a = 0$
ξ_1	$-\frac{p_b - \sqrt{p_b^2 - 4 \cdot p_a \cdot p_c}}{p_a}$	$p_a \neq 0 \wedge p_b^2 - 4 \cdot p_a \cdot p_c \geq 0$
ξ_2	$-\frac{p_b + \sqrt{p_b^2 - 4 \cdot p_a \cdot p_c}}{p_a}$	$p_a \neq 0 \wedge p_b^2 - 4 \cdot p_a \cdot p_c \geq 0$

The sign-invariant solution intervals cannot be determined as easily as for the univariate case because no ordering of the zeros is possible. Since we are only interested in an assignment that lies safely within each of the solution intervals, an ϵ-expression is used to describe an assignment as close as possible to the right of a zero and $-\infty$ can be used for the leftmost interval.

Definition 2.2.7. *A symbolic assignment α_{sym} or substitution is a function deified as follows:*

$$\alpha_{sym} : \{x_0, \ldots, x_n\} \to \{t_0, \ldots, t_n\}$$

where $\{x_0, \ldots, x_n\}$ denotes a set of variables and $\{t_0, \ldots, t_n\}$ is a set of substition terms, that can consist out of square-root expressions, ∞ and/or infinitesimals, that are possibly connected via addition.

Definition 2.2.8. *The set of symbolic assignments with $\alpha_{sym}(x) = t$ (and $\alpha_{sym}(y) = y$ for variables $y \neq x$) where*

$$t \in \{-\infty, \xi_0, \xi_0 + \epsilon, \xi_1 + \epsilon, \xi_2, \xi_2 + \epsilon\}$$

contains exactly one assignment for each sign-invariant solution interval of a polynomial $p = p_a \cdot x^2 + p_b \cdot x + p_c$ and will be denoted as rep(p,x).

For a given constraint $p \sim 0$ we only need to consider a certain subset of the symbolic assignments given by Definition 2.2.8 which depends on the relation symbol $\sim$. This is the case since we are interested in possible satisfying assignments of $p \sim 0$, e.g. a sign-invariant solution interval in which $\mathrm{sgn}(p) = 0$ is not suited to be a solution of a constraint of the shape $p < 0$.

Definition 2.2.9. *The test candidates tcs(cons,x) for a constraint $cons = p \sim 0$ and a variable x are defined as a set of symbolic assignments with $\alpha_{sym}(y) = y$ for all variables $y \neq x$ and $\alpha_{sym}(x) = t$ where*

$$t \in \{-\infty, \xi_0, \xi_1, \xi_2\} \text{ if } \sim \in \{\leq, \geq, =\} \text{ or}$$
$$t \in \{-\infty, \xi_0 + \epsilon, \xi_1 + \epsilon, \xi_2 + \epsilon\} \text{ if } \sim \in \{<, >, \neq\}$$

and every possible t appears in exactly one assignment. Each value for t has a side condition, where $sc(t) = sc(\xi_i)$ if $t = \xi_i$ or $t = \xi_i + \epsilon$ and $sc(t) = true$ otherwise.

Definition 2.2.10. *The test candidates $tcs(\varphi, x)$ for a real algebraic formula φ and a variable x are defined as:*

$$tcs(\varphi,x) = \bigcup_{cons \in Constraints(\varphi)} tcs(cons,x)$$

Note here that sometimes the abbreviated statements $t \in tcs(cons,x)$ or $t \in tcs(\varphi,x)$ are used, which serve as a short version of "there is a symbolic assignment $\alpha_{sym} \in tcs(cons,x)$ or $\alpha_{sym} \in tcs(\varphi,x)$ such that $\alpha_{sym}(x) = t$".

The important quantifier-elimination Theorem 2.2.1 can be deducted from this, which is proven by by Weispfenning [Wei97].

Theorem 2.2.1. *For any quantified formula $\exists x\varphi$, where $x \in free(\varphi)$ and φ fulfills the degree restriction of the virtual substitution for x, it holds that*

$$\exists x\varphi \equiv \bigvee_{t \in tcs(\varphi,x)} sc(t) \wedge \varphi[t//x]$$

where $\varphi[t//x]$ denotes the virtual substitution of x in φ by the term t.

Example 2.2.2. *We consider the example*

$$\vartheta := \exists x\varphi = \exists x(\underbrace{2 - x^2 - y^2 < 0}_{c_1} \wedge \underbrace{x^2 + y^2 - 5 < 0}_{c_2} \wedge \underbrace{x - 1 < 0}_{c_3} \wedge \underbrace{x + 1 > 0}_{c_4})$$

To determine $tcs(\varphi,x)$, we find the zeros of each constraint:

$$\begin{aligned} 2 - x^2 - y^2 \stackrel{!}{=} 0 \rightarrow \xi_{0,0} &= \sqrt{2 - y^2} \text{ if } 2 - y^2 \geq 0 \\ \xi_{0,1} &= -\sqrt{2 - y^2} \text{ if } 2 - y^2 \geq 0 \end{aligned}$$

$$\begin{aligned} x^2 + y^2 - 5 \stackrel{!}{=} 0 \rightarrow \xi_{1,0} &= \sqrt{5 - y^2} \text{ if } 5 - y^2 \geq 0 \\ \xi_{1,1} &= -\sqrt{5 - y^2} \text{ if } 5 - y^2 \geq 0 \end{aligned}$$

$$\begin{aligned} x - 1 \stackrel{!}{=} 0 \rightarrow \xi_{2,0} &= 1 \\ x + 1 \stackrel{!}{=} 0 \rightarrow \xi_{3,0} &= -1 \end{aligned}$$

In the next step we can use Definition 2.2.9 to find the terms t_i for $\alpha_{sym}(x) = t$ in $tcs(c_i,x)$.

$$\begin{aligned} tcs(c_1,x) &: \ t \in \{-\infty, \xi_{0,0} + \epsilon, \xi_{0,1} + \epsilon\} \\ tcs(c_2,x) &: \ t \in \{-\infty, \xi_{1,0} + \epsilon, \xi_{1,1} + \epsilon\} \\ tcs(c_3,x) &: \ t \in \{-\infty, \xi_{2,0} + \epsilon\} \\ tcs(c_4,x) &: \ t \in \{-\infty, \xi_{3,0} + \epsilon\} \end{aligned}$$

Applying Definition 2.2.10 and Theorem 2.2.1 yields:

$$tcs(\varphi,x) : \ t \in \{-\infty, \xi_{0,0} + \epsilon, \xi_{0,1} + \epsilon, \xi_{1,0} + \epsilon, \xi_{1,1} + \epsilon, \xi_{2,0} + \epsilon, \xi_{3,0} + \epsilon\}$$

$$\begin{aligned} \exists x\varphi \equiv\ & \varphi[-\infty//x] \\ & \vee\ \varphi[\sqrt{2 - y^2} + \epsilon//x] \wedge 2 - y^2 \geq 0 \\ & \vee\ \varphi[-\sqrt{2 - y^2} + \epsilon//x] \wedge 2 - y^2 \geq 0 \\ & \vee\ \varphi[\sqrt{5 - y^2} + \epsilon//x] \wedge 5 - y^2 \geq 0 \\ & \vee\ \varphi[-\sqrt{5 - y^2} + \epsilon//x] \wedge 5 - y^2 \geq 0 \\ & \vee\ \varphi[1 + \epsilon//x] \vee\ \varphi[-1 + \epsilon//x] \end{aligned}$$

2.2.3 Substitution rules

For the quantifier elimination to work, the remaining problem now is to solve the question how to handle the substitution of terms that contain ∞, ϵ and/or square-root expressions since the results of "naive" substitution would yield terms which themselves are not part of real arithmetic. Weispfenning [Wei97] gives a set of rules that cover all of these cases for up to quadratic polynomials. These rules have been simplified and implemented by Corzilius [Cor10][Cor16]. At this point we will revisit Example 2.2.2 to show some of these rules; the entire set of rules can be found in [Cor10].

Example 2.2.3. *Consider φ from Example 2.2.2. We execute the virtual substitution $\varphi[-1+\epsilon//x]$ (here only c_1 will be shown). For a polynomial $p_a x^2 + p_b x + p_c$ with $p_a \neq 0$ and substitution $t + \epsilon$, it is sufficient to test if the polynomial or its first or second derivative is negative for t:*

$$\begin{aligned}(2-x^2-y^2<0)[-1+\epsilon//x] \equiv\ & (2-x^2-y^2<0)[-1//x] \\ & \vee (2-x^2-y^2=0)[-1//x] \wedge (-2x<0)[-1//x] \\ & \vee (2-x^2-y^2=0)[-1//x] \wedge (-2x=0)[-1//x] \wedge (-1<0)[-1//x] \\ & \equiv 1-y^2<0\end{aligned}$$

The virtual substitution procedure can be described as a tree in which the root is the original formula and each edge depicts the virtual substitution of a variable labeled with the according term and its side condition. The constraint it emerges from is indicated by color. As soon as the substitution is complete (all quantified variables are substituted) or $true/false$ is the result, a leaf node is written. This is shown for $\exists x \exists y(\varphi)$ with φ from Example 2.2.2 in Figure 2.2.

2.3 SMT Solving

Since an SMT solver is an important part of the supposed algorithm in this thesis, a brief explanation what SMT solving is and how it works is perpended. The satisfiabilty problem for propositional logic is NP-complete. Nevertheless, a set of modern solvers exists that can solve satisfiabilty instances of propositional formulas with high efficiency. To extend the existing knowledge on more expressive logics such as the above mentioned (non-linear) real arithmetic, a combined approach of SAT solving (propositional logic) and theory solvers has been developed in the field of SMT solving. The general idea is to find a Boolean abstraction of the given first-order logic constraints by replacing them with Boolean variables, invoking a SAT solver to find a subset of these constraints that satisfies the Boolean abstraction and afterwards verify the consistency of these constraints within the theory. This approach is often termed as *lazy SMT solving*; further information on the techniques can be found in [ÁK17].

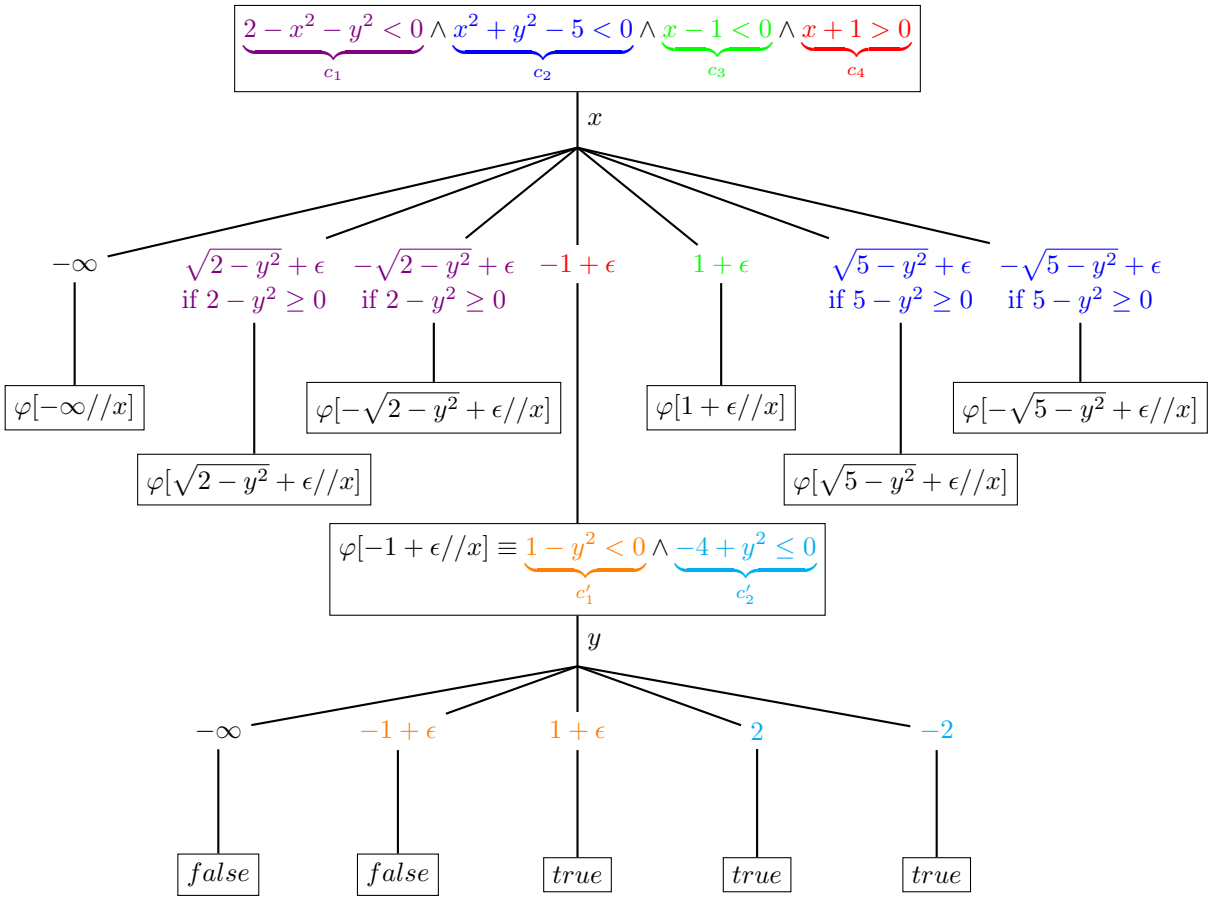

Figure 2.2: Tree description of the virtual substitution of $\exists x \exists y(\varphi)$. Note that the tree is not complete, since some branches have been not expanded to the end.

Chapter 3

Sample-based virtual substitution

3.1 General idea

As seen in Theorem 2.2.1 and also in Example 2.2.2, even for seemingly simple problems the complexity of the virtual substitution procedure is high. The reason for this is that a lot of test candidates and their respective substitutions have to be executed. Intuitively from the disjunction shown in Theorem 2.2.1, several test candidates might simply evaluate to $false$ if they do not represent a region a solution of the original formula can be found in. Nevertheless, the virtual substitution process still has to perform the virtual substitution of all these test candidates until - by simplification or direct result - the formula $false$ is returned.

Therefore, the idea proposed here is to only expand those branches of the virtual substitution tree that actually contain a solution, e.g. the value $true$ at one of their leaf nodes. This is supposed to reduce the complexity of the virtual substitution, since only a subset of all test candidates has to be considered. Whether the overall time complexity of the quantifier elimination procedure is reduced by this idea depends on the answer to the question whether the time complexity of the process of identifying which subset should be considered outweighs the simplification done to the virtual substitution or not.

In Figure 3.1, you can see a schematic description of the procedure. The input formula of the type $\exists x_0 \dots \exists x_n \varphi$ is entering the parser which then collects a *QEQuerry* (ordered list of quantified variables of input formula) and the quantifier-free formula. The latter is given to the SMT solver, which then tries to find a satisfying assignment for the quantifier-free formula. If the SMT solver returns SAT, the sample is passed to the sample-based virtual substitution module which invokes the virtual substitution restricted on the branch the current sample lies in. This results in a formula denoted as $\varphi_{\text{region}_i}$, where i denotes how often the SMT solver has run before. The negated $\varphi_{\text{region}_i}$ is added to the SMT solver (which still contains all previous formulas) while $\varphi_{\text{region}_i}$ will be remembered for the result. This loop continues until the SMT solver returns UNSAT. In this case, the disjunction of all $\varphi_{\text{region}_i}$ will be returned as the quantifier-free result of the quantifier elimination.

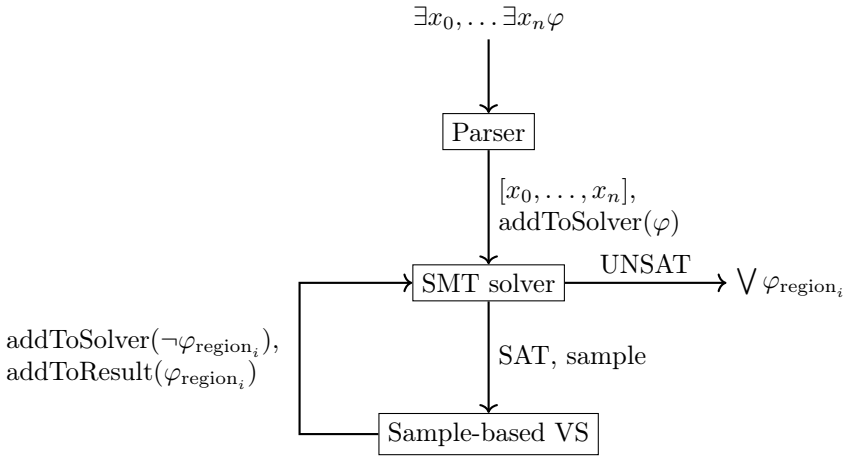

Figure 3.1: Schematic description of the proposed sampled-based virtual substitution procedure.

3.2 Algorithm

The cooperation with the SMT solver as described in the previous section is given in pseudocode in Algorithm 1. Here the QEQuerry denotes the set of quantified variables in the original formula, while inputFormula stands for the quantifier-free part of the original formula. In each step the invoked SMT solver tests the combination of the quantifier-free input formula and the previously found regions for satisfiability. This produces a sample if SAT has been found. The term "add" stands for a conjunction (with the previous content) in case something is added to the SMT solver and for a disjunction (with the previous content) in case something is added to the result.

Algorithm 1 Outer loop of sample-based virtual substitution.

```
1:  procedure QEWithSampleBasedVS(inputFormula, QEQuerry)
2:      initialize SMT solver
3:      add inputFormula to SMT solver
4:      result ← empty formula
5:      answer ← SMT solver check SAT
6:      while answer = SAT do
7:          sample ← sample from SMT solver
8:          tempFormula ← InputFormula
9:          for variable in QEQuerry do
10:             tempFormula ← EliminateVar(tempFormula,variable, sample)
11:         region ← tempFormula
12:         add negated region to SMT solver
13:         add region to result
14:         answer ← SMT solver check SAT
15:     return result
```

Algorithm 2 shows the inner procedure of this quantifier elimination approach. It receives the original input formula, a variable that should be eliminated and a sample that has been generated by the SMT solver. For each constraint the symbolic zeros are determined of which some will be removed, namely the ones where the side condition is not *true* under the given sample. After that, all remaining zeros are evaluated under the sample and the resulting numerical values are sorted. To determine in which of the now formed intervals the sample lies, the value of sample(variable) is assigned to its position in the ordering. For the sample α in theory three cases appear:

1. α(variable) is smaller than all numeric zeros

2. α(variable) is exactly the value of one of the appearing numeric zeros $[\![\xi]\!]^\alpha$.

3. Sample(variable) lies between two numeric zeros $[\![\xi_a]\!]^\alpha$ and $[\![\xi_b]\!]^\alpha$. ξ_a and ξ_b are the closest bounds for α(variable), e.g. there is no ξ'_a with

   $$[\![\xi_a]\!]^\alpha < [\![\xi'_a]\!]^\alpha < \alpha(\text{variable})$$

   and no ξ'_b with

   $$\alpha(\text{variable}) < [\![\xi'_b]\!]^\alpha < [\![\xi_b]\!]^\alpha.$$

We then want to reconstruct a term t from $rep(\varphi, x)$ that is associated to the region the sample lies within (line 8). To do so, we once again consider the above-mentioned cases:

1. Chose $t = -\infty$

2. Chose $t = \xi$

3. Chose $t = \xi_a + \epsilon$

We note here that some of the chosen terms might not appear in the set of test candidates as defined in Definition 2.2.9 and we will show later on that this does not result in a restriction of correctness.

The virtual substitution is then performed on the input formula and variable using the previously determined term. Another simplification is done here: The resulting formula of the virtual substitution (according to [Cor10]) is in most cases a disjunction of mutually exclusive formulas. By evaluating these formulas using the sample, we identify the only part of the disjunction that is satisfied by the sample and only return that as the result. This can be seen as an internal simplification step: In case a part of the disjunction that is unsatisfiable is omitted, it does not matter for the final result of quantifier elimination anyway. In case it can be satisfied (and is not covered by another term), the algorithm will produce another sample that satisfies it and the virtual substitution of this term will be revisited again and this part of the disjunction will be collected to the result. Note here that it is not tested whether the omission of these disjunctive formulas increases the overall performance or if the possible revisiting of a term actually increases the overall cost.

Algorithm 2 Algorithmic structure of inner elimination method.

```
1: procedure EliminateVar(inputFormula, variable, sample)
2:     determine symbolic zeros and side conditions of inputFormula with variable
3:     check side conditions with sample
4:     remove zeros with unsatisfied side condition
5:     numericZeros ← evaluate zeros with sample
6:     order numericZeros
7:     termNum ← biggest value in numericZeros ≤ sample(variable)⋆
8:     term ← backtrack symbolic term of termNum
9:     result ← virtualSubstitution(inputFormula, variable, term)
10:    return result
```

⋆: In case no value from numericZeros is smaller or equal, skip this step and set term $\leftarrow -\infty$ in line 8.

Definition 3.2.1. *We define $rep'(\varphi,x) \subseteq rep(\varphi,x)$ as the set of representatives that are determined by Algorithm 1. Precisely $rep'(\varphi,x)$ is the set of symbolic assignments α_{sym} with $\alpha_{sym}(x) = t$ and t is one of the terms found in Algorithm 2 (line 8), when Algorithm 1 is applied on a formula of the type $\exists x(\varphi)$ with a quantifier-free φ.*

Theorem 3.2.1. *For any quantified formula $\exists x\varphi$, where $x \in free(\varphi)$ and φ fulfills the degree restriction of the virtual substitution for x, it holds that*

$$\exists x\varphi \equiv \bigvee_{t \in rep'(\varphi,x)} sc(t) \wedge \varphi[t//x].$$

3.3 Example

Example 3.3.1. *We revisit the formula*

$$\vartheta := \exists x\varphi = \exists x(\underbrace{2 - x^2 - y^2 < 0}_{c_1} \wedge \underbrace{x^2 + y^2 - 5 < 0}_{c_2} \wedge \underbrace{x - 1 < 0}_{c_3} \wedge \underbrace{x + 1 > 0}_{c_4})$$

from Example 2.2.2. The satisfying region of this problem is shown in green in Figure 3.2. Since there are solutions for φ a sample will be produced, here the sample α_1 with $\alpha_1(x) = 0$ and $\alpha_1(y) \approx -1.618$ is determined. The symbolic zeros including their side condition of the constraints are calculated (similar to Example 2.2.2), which yields the set

$$\{\pm\sqrt{2-y^2} \text{ if } 2 - y^2 \geq 0, \pm\sqrt{5-y^2} \text{ if } 5 - y^2 \geq 0, \pm 1 \text{ if } true\}$$

Since $[\![2 - y^2 \geq 0]\!]^{\alpha_1} \equiv false$, the side condition of the zeros $\pm\sqrt{2-y^2}$ is not satisfied by α_1 and they are removed for this iteration of the loop. The evaluation of the remaining symbolic zeros yields the set of numeric zeros as:

$$\{-1.54, -1, +1, +1.54\}$$

Since $\alpha_1(x) = 0$ the value lies between -1 and 1 as the closest numeric zeros from below and above respectively, therefore $t_1 = -1 + \epsilon$ is chosen and $t_1 \in rep'(\varphi, x)$

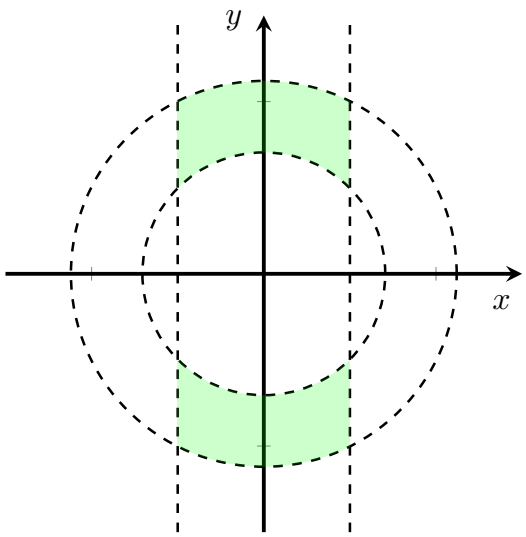

Figure 3.2: Plot of satisfying region of φ from Example 2.2.2. The constraints are shown dashed (since the all have relations $<$ or $>$) and the satisfying region is denoted in green.

is assigned. The virtual substitution $\varphi[t//x]$ *produces true for* $c_3[t//x]$ *and* $c_4[t//x]$ *while for* c_1 *and* c_2 *a disjunction of three terms is obtained from which (according to the simplification described above) only that part is kept which is satisfied by* α_1*. Therefore*

$$c_1[t//x] = 1 - y^2 < 0 \text{ and } c_2[t//x] = -4 + y^2 < 0.$$

Note here that according to Example 2.2.3, two of three of the terms in the disjunction evaluate to $false$ *anyway during the substitution of* c_1 *while for the substitution of* c_2 *a term that is satisfiable is omitted due to this simplification. The resulting region of the first loop iteration is*

$$\varphi_{region_1} = 1 - y^2 < 0 \wedge -4 + y^2 < 0.$$

The relation between the entire solution space and φ_{region_1} *is displayed in Figure* 3.3*.*

φ_{region_1} *is subsequently excluded from the search for possible samples of* φ*. The next sample found is* α_2 *with* $\alpha_2(x) = 0$ *and* $\alpha_2(y) = -2$*. The same process as described above is carried out and yields* $t_2 = -\sqrt{5 - y^2} + \epsilon \in rep'(\varphi, x)$*. The simplified virtual substitution is performed, which yields* $true$ *for* c_1 *and* c_3*. For* c_2 *and* c_4*, the simplification restricts us to one part of the resulting disjunction that is satisfied by* α_2 *and in conclusion* φ_{region_2} *is obtained as*

$$\varphi_{region_2} = y^2 - 5 < 0 \wedge -4 + y^2 = 0.$$

The graphical description of φ_{region_2} *is shown in Figure 3.4.*

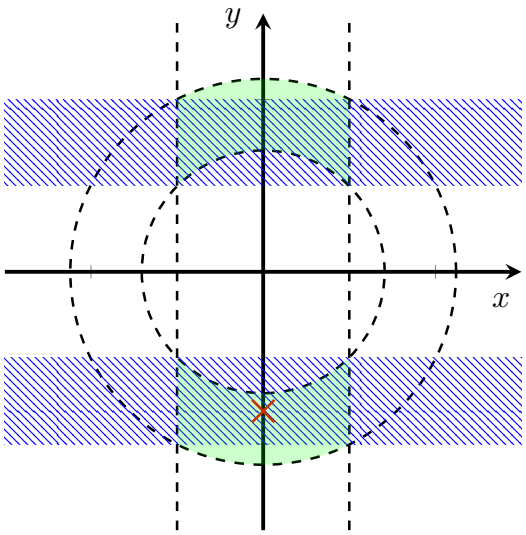

Figure 3.3: Plot of $\varphi_{\text{region}_1}$ (blue, dashed) determined from φ from Example 2.2.2. The satisfying region is shown in green. α_1 is displayed as a red cross.

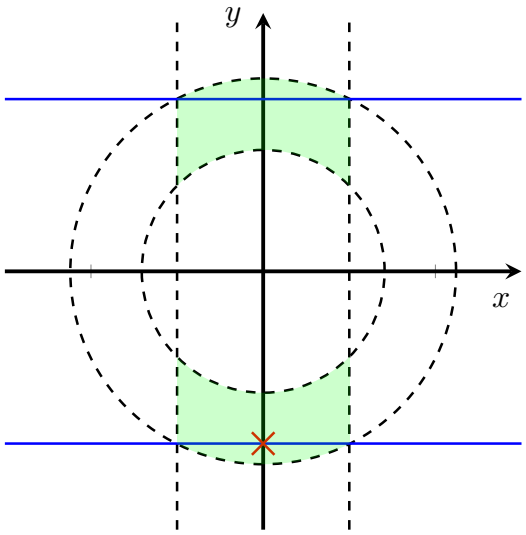

Figure 3.4: Plot of $\varphi_{\text{region}_2}$ (blue line) determined from φ from Example 2.2.2. The satisfying region of φ is shown in green. α_2 is displayed as a red cross.

For the final iteration of the loop, t_2 is found again from the third sample α_3 with $\alpha_3(x) = 0$ and $\alpha_3(y) = -2.12$. Note here that the aforementioned case occurs where the same term has to be revisited due to the simplification during the virtual substitution. The simplified virtual substitution still yields true for c_1 and c_3 and disjunctions of multiple constraints for c_2 and c_4. Due to the differing sample, a different constraint is picked from the substitution result of c_4 resulting in

$$\varphi_{region_3} = y^2 - 5 < 0 \wedge 4 - y^2 < 0$$

which is shown in Figure 3.5.

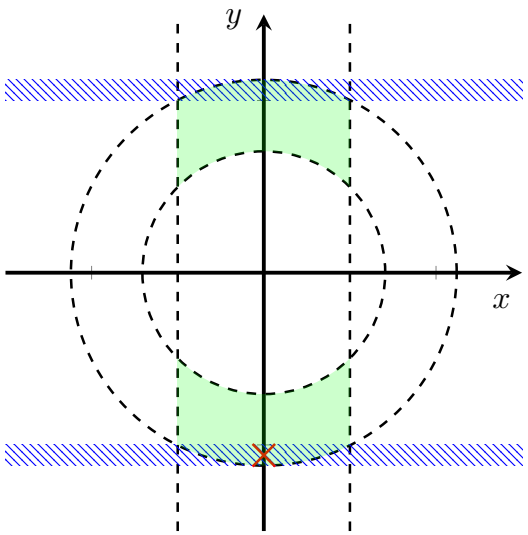

Figure 3.5: Plot of $\varphi_{\text{region}_3}$ (blue,dashed) determined from φ from Example 2.2.2. The satisfying region of φ is shown in green. α_3 is displayed as a red cross.

The three formulas found cover the entire solution space of φ in y-direction and are therefore a suitable result of quantifier elimination. The SMT solver consequently returns UNSAT after the third iteration and the result is given by:

$$\exists x\varphi \equiv \varphi_{region_1} \vee \varphi_{region_2} \vee \varphi_{region_3}$$

The process is summarized in Table 3.1.

Iteration i	$t \in rep'(\varphi,x)$	$\varphi_{\text{Region}_i}$
1	$-1+\epsilon$	$1-y^2<0 \wedge -4+y^2<0$
2	$-\sqrt{5-y^2}+\epsilon$	$y^2-5<0 \wedge -4+y^2=0$
3	$-\sqrt{5-y^2}+\epsilon$	$y^2-5<0 \wedge 4-y^2<0$

Table 3.1: Iteration steps for quantifier elimination of $\exists x\varphi$.

3.4 Correctness

Lemma 3.4.1. *Let $\alpha_{sym} \in rep'(\varphi,x)$ be a symbolic assignment with $\alpha_{sym}(x) = t$ and $\alpha_{sym} \notin tcs(\varphi,x)$. For any assigment β it holds that*

$$\beta \vDash sc(t) \wedge \varphi[t//x] \rightarrow \beta \vDash \bigvee_{t' \in tcs(\varphi,x)} \varphi[t'//x] \wedge sc(t').$$

Proof. Proof of Lemma 3.4.1. Let β be the assignment as given in the lemma. We have a look at the set $rep'(\varphi,x)$ that has been determined from the algorithm. It is obvious that

$$rep'(\varphi,x) \subseteq \bigcup_{cons \in \varphi} rep(cons,x)$$

holds, since the only results for $rep'(\varphi,x)$ generated by the algorithm are per construction symbolic assignments with $\alpha_{\text{sym}}(x) = t$ and t being $-\infty$, the zeros of p for $p \sim 0 \in cons(\varphi)$ or one of those zeros plus an infinitesimal.

Now we have to consider the relationship between $rep'(\varphi,x)$ and $tcs(\varphi,x)$: We assume that there is a term t so that $t \in rep'(\varphi,x) \setminus tcs(\varphi,x)$. Two cases have to be taken into account:

1. $t = \xi_i + \epsilon$, where ξ_i is the (symbolic) zero of a polynomial p that appears in φ as the constraint $p \sim 0$ with $\sim \in \{\leq, \geq, =\}$ (compare Definition 2.2.9).

   It is also given from the condition of this lemma that $\beta \vDash \varphi[t//x] \wedge sc(t)$. We now show that

   $$\beta \vDash \bigvee_{t' \in tcs(\varphi,x)} \varphi[t'//x] \wedge sc(t')$$

   exemplary with one of the substitution rules given by [Cor10].

   We consider a single constraint $cons = l < 0$ with $l = a \cdot x^2 + b \cdot x + c$ from φ. Note that $cons$ is a different constraint in φ from the constraint that produced t in the algorithm. We want to show

   $$\beta \vDash cons[t//x] \wedge sc(t) \rightarrow \bigvee_{t' \in tcs(\varphi,x)} cons[t'//x] \wedge sc(t')$$

   The substitution $cons[t//x] \equiv cons[\xi_i + \epsilon//x]$ yields

   $$\begin{aligned} cons[\xi_i + \epsilon//x] \equiv & \underbrace{((a \cdot x^2 + b \cdot x + c < 0)[\xi_i//x])}_{c_1} \\ & \vee \underbrace{((a \cdot x^2 + b \cdot x + c = 0)[\xi_i//x] \wedge 2a \cdot x + b < 0[\xi_i//x])}_{c_2} \\ & \vee \underbrace{((a \cdot x^2 + b \cdot x + c = 0)[\xi_i//x] \wedge (2a \cdot x + b = 0)[\xi_i//x] \wedge (2a < 0)[\xi_i//x])}_{c_3} \end{aligned}$$

   From $\beta \vDash cons[t//x]$ we conclude $\beta \vDash c_1 \vee c_2 \vee c_3$. We know that $t' \in tcs(\varphi, x)$ for $t' = \xi_i$ (see $p \sim 0 \in \varphi$ above). c_1 therefore also appears in $\bigvee_{t' \in tcs(\varphi,x)} cons[t'//x] \wedge sc(t')$, which proves the statement for $\beta \vDash c_1$. In case $\beta \vDash c_2 \vee c_3$ we have $\beta \vDash (a \cdot x^2 + b \cdot x + c = 0)[\xi_i//x]$. That implies ξ_i is *also* a zero of l. According to Theorem 2.2.9, this means that $\xi_i + \epsilon \in tcs(\varphi,x)$ which contradicts the assumption from this lemma ($t \in rep'(\varphi,x) \setminus tcs(\varphi,x)$).

   Extending this kind of argument on different substitution rules proves the lemma for any set of constraints φ in this sub-case with $t = \xi_i + \epsilon$.

2. $t = \xi_i$, where ξ_i is the (symbolic) zero of a polynomial p that appears in φ as the constraint $p \sim 0$ with $\sim \in \{<, >, \neq\}$. Since the according solution interval

to $t = \xi_i$ only contains ξ_i it follows from the algorithm that for the considered sample α the statement

$$\alpha(x) = [\![\xi_i]\!]^\alpha$$

holds. Otherwise the algorithm would not have assigned t to a sample and t would not appear in $rep'(\varphi,x)$. This means that $[\![p]\!]^\alpha = 0$ and consecutively $[\![p \sim 0]\!]^\alpha \equiv false$. α is therefore not a sample, contradiction. This case will therefore never occur in $rep'(\varphi,x)$.

Since case 2 never occurs and in case 1 it has been shown that the lemma holds, this proves the lemma. □

Proof. Proof of Theorem 3.2.1. For this proof we assume a "perfect" SMT solver, i.e. one that returns SAT and a valid sample if the given formulas are satisfiable. If not, it returns UNSAT.

We consider the quantifier-elimination of $\exists x\varphi$. It is enough to prove that

$$\underbrace{\bigvee_{t \in rep'(\varphi,x)} \varphi[t//x]}_{\vartheta} \equiv \underbrace{\bigvee_{t \in tcs(\varphi,x)} \varphi[t//x]}_{\vartheta'}$$

because from this the desired theorem immediately follows with Theorem 2.2.9. We assume the contrary, so either there is an assignment α with $\alpha \vDash \vartheta$ and $\alpha \nvDash \vartheta'$ or an β with $\beta \nvDash \vartheta$ and $\beta \vDash \vartheta'$.

Case 1: Assume there is an assignment α with α with $\alpha \vDash \vartheta$ and $\alpha \nvDash \vartheta'$. If $\alpha \vDash \vartheta$ there is at least one $t \in rep'(\varphi,x)$ such that $\alpha \vDash \varphi[t//x]$. If $t \in rep'(\varphi,x) \cap tcs(\varphi,x)$, $\varphi[t//x]$ also appears in ϑ', therefore $\alpha \vDash \vartheta'$, contradiction. The remaining case is therefore $t \in rep'(\varphi,x) \setminus tcs(\varphi,x)$, which is covered by Lemma 3.4.1. Here we also get $\beta \vDash \vartheta'$, a contradiction. There is no α with the given properties.

Case 2: Assume there is an assignment β with β with $\beta \nvDash \vartheta$ and $\beta \vDash \vartheta'$. From Theorem 2.2.9 it follows that $\beta \vDash \exists x\varphi$. We create an extension of β', which assigns a value to x. Due to $\beta \vDash \exists x\varphi$ we can always chose a c and $\beta'(x) = c$ such that $\beta' \vDash \varphi$. Since x does not appear in ϑ using the coincidence lemma and $\beta \nvDash \vartheta$ we get $\beta' \nvDash \vartheta$. From $\beta' \nvDash \vartheta$ and $\beta' \vDash \varphi$ we conclude $\beta' \vDash (\neg\vartheta \wedge \varphi)$. The SMT solver should therefore return SAT and β' (or another sample), if executed on $\neg\vartheta \wedge \varphi$. This is a contradiction, since for ϑ to be the result of the algorithm, the last answer should have been UNSAT. Therefore there is no assignment β with the given properties.

From these two cases we can conclude that Theorem 3.2.1 holds and the algorithm works correctly. Note that for this proof we have omitted the part of the algorithm that may discard some parts of a substitution result to avoid disjunctions. Therefore it is possible that a $t \in rep'(\varphi,x)$ has to be visited multiple times (also shown in the previous example). This however does not influence the correctness due to similar arguments as shown in this proof. □

Chapter 4

Experimental Results

The presented algorithm has been implemented within the SMT-RAT framework. For the SMT solver, the "NRA-solver" strategy has been chosen which was modified by removing the virtual substitution and the preprocessing module. The current implementation restricts the input formula to be a conjunction of constraints. Testing has been done on several simple examples that have been crafted for testing or were given by Geogebra developers. The command line output for the Example 2.2.2 can be seen in Figure 4.1. It fits the expectations for the result.

```
$ ./smtrat-shared Example1.txt
Quantified Formula: QE(exists x) (-1 + x < 0 and -5 + x^2 + y^2 < 0 and 2 + -1*x^2 + -1*y^2 < 0 and -1 + -1*x < 0)
Current Sample is: {x : 0, y : -910872158600853/562949953421312}
Current Term is: -1 + epsilon
Region found: (-4 + y^2 < 0 and 1 + -1*y^2 < 0)
Current Sample is: {x : 0, y : -2}
Current Term is: 0+-1*sqrt(5 + -1*y^2) + epsilon
Region found: (-5 + y^2 < 0 and -5 + y^2 <= 0 and -4 + y^2 = 0)
Current Sample is: {x : 0, y : -1192347135311509/562949953421312}
Current Term is: 0+-1*sqrt(5 + -1*y^2) + epsilon
Region found: (-5 + y^2 < 0 and -5 + y^2 <= 0 and 4 + -1*y^2 < 0)
Equivalent Quantifier-Free Formula: ((-4 + y^2 < 0 and 1 + -1*y^2 < 0) or (-5 + y^2 < 0 and -5 + y^2 <= 0 and -4 + y^2 = 0)
or (-5 + y^2 < 0 and -5 + y^2 <= 0 and 4 + -1*y^2 < 0))
```

Figure 4.1: Command line output when the quantifier elimination is executed on Example1.txt, which constraints 2.2.2 in SMT-LIB syntax.

However, no sufficient verification of the implementation and speed comparison has been done. Some technical problems in handling of certain real algebraic numbers that cannot be converted to rationals, such as $\sqrt{2}$, have occurred and need to be solved in order for an extensive run time comparison study on a set of more complex problems. The remaining problems and possible solutions are explained in the next chapter.

Chapter 5

Conclusion

5.1 Summary

In this thesis, a quantifier elimination approach has been discussed and implemented. The approach is based on virtual substitution, simplified by using a sample-based procedure which makes use of existing SMT solvers for sample generation. From this, a reduction in overall complexity of the standard virtual substitution is expected. The correctness of the method has been proven and the algorithm was implemented within the SMT-RAT framework, but still needs some modifications to be broadly applicable.

5.2 Discussion

A complete analysis of the current implementation is difficult since concrete results on the run time of the algorithm in comparison to existing implementations are still missing. Nevertheless, this methods promises a substantial reduction of complexity for the virtual substitution and has been proven to be correct.

The interesting question to answer is whether or not the reduction of complexity during the virtual substitution outweighs the increase effort to call an SMT solver several times during the procedure. We will list the remaining problems to be solved in order for the method to be applicable for arbitrary input formulas and in order to produce insightful benchmarks in the next section.

5.3 Future work

Implementation of quantifier elimination by standard virtual substitution
The standard virtual substitution for quantifier elimination should be implemented within the SMT-RAT framework in order compare the efficiency with the method proposed in this thesis.

Conversion into prenex normal form (PNF)
Since the proposed algorithm works with a formula that is in prenex normal form, an arbitrary input formula should be converted in PNF beforehand. Efficient algorithms for this, which include renaming variables and rearranging negations, exist. However,

they are not implemented in the SMT-RAT framework so far.

Universal quantifiers
As of right now, an universal quantifier in the input formula is not permitted and the procedure will be terminated. Since a double negation can convert an universal quantifier into an existential quantifier as follows

$$\forall x \varphi \equiv \neg \exists x \neg \varphi,$$

covering universal quantifiers using that rule should be implemented in the future.

Verification of the implementation
To verify the implementation, a test study and comparison with results of a tool that allows for quantifier elimination such as Z3 [dMB08] should be done. We suggest the usage of SMT-LIB examples from the QF-NRA section in which a random subset of variables is chosen to be quantified in the tested problems. After the verification, a run time comparison should be carried out to compare the existing implementation with the new approach.

Handling of non-ration real algebraic numbers (RAN)
An important step to make this method broadly applicable is the handling of non-rational RAN such as $\sqrt{2}$ during the algorithm. In the current implementation they are stored as $(p,(a,b))$, where $p \in \mathbb{R}[x]$, $a,b \in \mathbb{R}$ and $a < b$. The described RAN is a zero of the given polynomial p that lies within the interval given by (a,b). As of right now, the main part of the algorithm (compare Algorithm 2) exclusively works with rational numbers in the sample. As soon as a non-rational RAN appears in one of the samples that have been found, the algorithm cannot proceed. To solve this problem, the algorithm has to be modified to be able to evaluate symbolic zeros using the description of RAN using a polynomial and an interval. Therefore, the numeric zeros that are found may also be represented in the fashion described above for RAN. Since non-rational RAN still allow for ordering, this is merely a technical problem rather than a problem of theoretical nature.

Validation of simplification
As described in the main part, there is a simplification done after each virtual substitution that only selects one part of the resulting disjunction, namely the part that is satisfied by the sample. This reduces the complexity of the resulting regions, but may also lead to revisiting a term for virtual substitution that has been visited in a previous iteration of the algorithm. To check whether or not the average cost of possible revisiting exceeds the gain of simplification, a run time comparison with an implementation that does not perform this step should be performed. Once again, here we suggest a similar set of test cases as described above.

Variation of SMT solving strategy
After an implementation that can be benchmarked and compared to Z3 successfully has been established, another approach is to vary the SMT solver involved in the procedure. We suspect that the sample-based quantifier elimination creates a special subset of SMT solving problems, namely subsequently finding solution regions from the same formula and consequently excluding them. Therefore, certain SMT solving strategies might be more efficient in handling such a problem structure, e.g. those

that have a high degree of incrementality might profit from the inherently incremental problem structure. Thus, we suggest a testing and comparison of different SMT solvers within this method.

Bibliography

[ÁK17] Erika Ábrahám and Gereon Kremer. SMT solving for arithmetic theories: Theory and tool support. In *2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 1–8. IEEE, 2017.

[Col75] George E Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Automata theory and formal languages*, pages 134–183. Springer, 1975.

[Cor10] Florian Corzilius. Virtual substitution in SMT solving. Master's thesis, RWTH Aachen University, 2010.

[Cor16] Florian Corzilius. *Integrating Virtual Substitution into Strategic SMT Solving.* PhD thesis, RWTH Aachen University, 2016.

[dMB08] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Stu17] Thomas Sturm. A survey of some methods for real quantifier elimination, decision, and satisfiability and their applications. *Mathematics in Computer Science*, 11(3):483–502, 2017.

[Tar49] Alfred Tarski. A decision method for elementary algebra and geometry. *Journal of Symbolic Logic*, 14(3):188–188, 1949.

[Wei97] Volker Weispfenning. Quantifier elimination for real algebra — the quadratic case and beyond. *Applicable Algebra in Engineering, Communication and Computing*, 8(2):85–101, 1997.