# New Concepts for Real Quantifier Elimination by Virtual Substitution

**Marek Košta**

**Dissertation**

zur Erlangung des Grades des
Doktors der Naturwissenschaften (Dr. rer. nat.)
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

Saarbrücken
August, 2016

| | |
|---|---|
| Tag des Kolloquiums: | 13.12.2016 |
| Dekan: | Prof. Dr. Frank-Olaf Schreyer |
| **Prüfungsausschuss** | |
| Vorsitzender: | Prof. Dr. Sebastian Hack |
| Berichterstatter: | PD Dr. Thomas Sturm |
| | Prof. Dr. Andreas Weber |
| | Prof. Dr. Christoph Weidenbach |
| Akademischer Mitarbeiter: | Dr. Swen Jacobs |

# Abstract

Quantifier elimination methods for real closed fields are an intensively studied subject from both theoretical and practical points of view. This thesis studies quantifier elimination based on virtual substitution with a particular focus on practically applicable methods and techniques. We develop a novel, standalone, and modular quantifier elimination framework for virtual substitution that can in principle be extended to arbitrary but bounded degrees of quantified variables. The framework subsumes previous virtual substitution algorithms. Quantifier elimination algorithms are obtained via instantiation of our quantifier elimination algorithm scheme with three precisely specified subalgorithms. We give instantiations of our scheme up to degree three of a quantified variable, which yields a quantifier elimination algorithm by virtual substitution for the cubic case. Compared to previous virtual substitution-based approaches, we propose novel improvements like smaller elimination sets and clustering.

Furthermore, we exploit the Boolean structure and develop a structural quantifier elimination algorithm scheme. This allows us to take advantage of subformulas containing equations or negated equations, simplify virtual substitution results, and develop flexible bound selection strategies. We also revisit the established technique of degree shifts and show how to generalize this within our structural quantifier elimination algorithm scheme.

Restricting ourselves to existential problems, we address the established concept of extended quantified elimination, which in addition to quantifier-free equivalents yields answers for existentially quantified variables. We show how to realize this concept within our quantifier elimination algorithm scheme. Moreover, we generalize our post-processing method for eliminating nonstandard symbols from answers to the general case.

Our implementation of most of the concepts developed in this thesis is the first implementation of a cubic virtual substitution method. Experimental results comparing our implementation with the established original implementation of the quadratic virtual substitution in the Redlog computer logic system demonstrate the relevance of our novel techniques: On more than two hundred quantifier elimination problems—considered in more than sixty scientific publications during the past twenty years—we never eliminate fewer quantifiers than the Redlog's original implementation. For a considerable number of problems we eliminate more quantifiers.

# Zusammenfassung

Quantoreneliminationsverfahren für reelle abgeschlossene Körper sind sowohl von der theoretischen als auch von der praktischen Seite ein intensiv studiertes Thema. Diese Dissertation befasst sich mit Quantorenelimination basierend auf virtueller Substitution. Im Mittelpunkt stehen praktisch anwendbare Methoden und Techniken. Wir entwicklen ein neues, unabhängiges und modulares Quantoreneliminationsrahmenkonzept für virtuelle Substitution, das im Prinzip auf beliebige Grade von quantifizierten Variablen erweitert werden kann. Unser Rahmenkonzept subsumiert existierende auf virtueller Substitution beruhende Algorithmen. Konkrete Algorithmen enstehen als Instanzen unseres Quantoreneliminationsalgorithmusschemas mit drei genau spezifizierten Subalgorithmen. Wir präsentieren Instanzen bis zu Grad drei einer quantifizierten Variable. Die liefern einen Algorithmus beruhend auf virtueller Substitution für den kubischen Fall. Im Vergleich mit anderen Verfahren basierend auf virtueller Substitution präsentieren wir zahlreiche Verbesserungen wie etwa kleinere Eliminationsmengen oder Clustering.

Außerdem nutzen wir die Boolsche Struktur aus und entwickeln ein strukturelles Quantoreneliminationsalgorithmusschema. Somit können wir Gleichungen oder negierte Gleichungen ausnutzen, Ergebnisse der virtuellen Substitution vereinfachen und flexible Schrankenauswahlstrategien entwickeln. Wir studieren auch die bekannte Technik des „degree shifts“, die in manchen Fällen den Grad der quantifizierten Variablen reduzieren kann. Wir zeigen wie man diese Technik in unserem Quantoreneliminationsalgorithmusschema realisiert und verallgemeinert.

Für reelle existentielle Probleme diskutieren wir das Konzept der erweiterten Quantorenelimination, die zu quantorenfreien Äquivalenten auch Antworten für die quantifizierten Variablen liefert. Wir zeigen wie sich dieses Konzept in unserem Quantoreneliminationsalgorithmusschema realisieren lässt. Zusätzlich verallgemeinern wir unser Postprocessingverfahren zur Elimination von Nichtstandardsymbolen aus Antworten.

Unsere Implementierung unterstützt die meisten in dieser Arbeit vorgestellte Konzepte und stellt damit die erste Implementierung einer kubischen Methode basierend auf virtueller Substitution dar. Praktische Rechenexperimente, in denen wir unsere Implementierung mit bekannten im Computerlogik-System Redlog implementierten Verfahren für quadratische virtuelle Substitution verglichen, zeigen die Relevanz unserer Techniken: Auf mehr als 200 in mehr als sechzig wissenschaftlichen Publikationen beschriebenen Quantoreneliminationsproblemen eliminiert unsere Implementierung niemals weniger Quantoren als die existierende Implementierung in Redlog. Für eine signifikante Anzahl von Problemen können wir sogar mehr Quantoren eliminieren.

# Acknowledgments

First and foremost I would like to thank my doctoral advisor Thomas Sturm. His intuition combined with an extreme sense for detail, precision, and formal rigor were attributes I learned from during every single discussion we had. While working with him I also had the opportunity to learn something from his programming mastery and his deep understanding of practical and technical aspects of the implementation process. Furthermore, the motto that describes Thomas very well is "Know your tools!" Understanding the importance of this motto was a great benefit that has deeply influenced my working style.

I am also indebted to my former advisor and the head of our group Christoph Weidenbach, who stimulated my interest in logic and invited me to his group. Without him I would have never gotten the opportunity to work in such an exciting field on the boundary of mathematics, logic, and computer science combining theory and practice.

Many thanks go also to my coauthor Andreas Dolzmann. Discussions we had together with Thomas were a great source of inspiration that directly influenced both the content and the structure of this thesis. Almost every question Andreas asked pointed to a subtle but substantial fact and kept me busy with answering; sometimes for minutes but sometimes for days.

I would also like to thank all the colleagues of the Automation of Logic group for a friendly and stimulating atmosphere. Special thanks go to Marco Voigt for proofreading parts of this dissertation.

Furthermore, I thank all the researchers in the AVACS project, especially in the subproject H1/2. Our regular meetings, discussions, and questions gave me the opportunity to view their and my own work from a completely different perspective, which was often very beneficial.

Many thanks go to my family and friends in Slovakia for their constant support and pleasant distractions from work during the regular vacation visits at home.

Last but not least I thank my wife Toňa: Without your constant love and support during the last five years, especially during the tough times in 2014 and 2015, nothing of this would have been possible.

This work has been supported by the International Max Planck Research School for Computer Science, by the German Transregional Collaborative Research Center SFB/TR 14 AVACS, and by the ANR/DFG Programme Blanc Project STU 483/2-1 SMArT.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Since the breakthrough results by Gödel [39] and Turing [79], the concepts of Turing machines, computability, and decidability have gradually become central concepts of modern mathematics, logic, and computer science. On the one hand their results showed that Hilbert's program was an unreachable goal in general. On the other hand they triggered an intense work on studying the decidability of various mathematical and algebraic theories, to accomplish at least partially the goals of Hilbert's program. A number of algebraic theories has played a central role in this endeavor from the beginning. In fact, Presburger's famous result [60] from 1929 on the completeness of what is today known as "Presburger Arithmetic" implied the decidability of the theory he considered even before the term was coined in the works of Gödel and Turing.

Quantifier elimination (QE) served as a key method for obtaining completeness and thus decidability results right from the beginning. Indeed, Presburger's result was a constructive proof, which yielded a quantifier elimination method as well. Another prominent and well-known algebraic theory, whose completeness and decidability was proven by providing a quantifier elimination algorithm, is the theory of real closed fields. In his breakthrough work [78] from 1948, Tarski gave a quantifier elimination algorithm for that theory, which plays a central role for geometry, physics, and the sciences in general.

Since the study of quantifier elimination methods for the theory of real closed fields is the main subject of this thesis, we give a few technical details already here. The Tarski language is given by $\mathfrak{L} = (0, 1, +, -, \cdot, \neq, <, \leq, \geq, >)$. The symbol "=" and its interpretation as equality are considered part of interpreted first-order logic so that it does not occur in $\mathfrak{L}$. The constants, function symbols, and relation symbols of $\mathfrak{L}$ are interpreted in the Tarski algebra $(\mathbb{R}, 0, 1, +, -, \cdot, \neq, <, \leq, \geq, >)$ with their usual meaning. The theory of real closed fields is the first-order theory of that Tarski Algebra.

A countable axiomatization of real closed fields [55, Chapter 8] can be obtained by adding to the theory of ordered fields the following axioms:

1. an axiom asserting that every positive number has a square root, formally: $\exists x\big(y > 0 \longrightarrow x^2 = y\big)$ and

2. an axiom scheme asserting that every polynomial of odd degree has at least one root, formally: $\exists x\big(c_{2n+1} \neq 0 \longrightarrow c_{2n+1}x^{2n+1} + \cdots + c_1x + c_0 = 0\big)$ for each $n \in \mathbb{N} \cup \{0\}$.

Besides its nice logical and model-theoretic properties, the theory of real closed fields turned out to be of great practical importance in the sciences and engineering. A giant step towards the practical applicability of the theory was made by Collins [21] in 1975. His doubly exponential Cylindrical Algebraic Decomposition (CAD) algorithm for real quantifier elimination was a tremendous improvement over Tarski's non-elementary algorithm. During the following decades, a number of improvements [22, 52] were proposed by Collins and his students, and a number of CAD-based algorithms [10, 18] and variants [14, 16] have been developed. Despite the doubly exponential worst-case complexity, efficient implementations of CAD [13, 66] have been successfully applied in the sciences and engineering [42, 15].

The inherent time complexity of the problem of real decision and quantifier elimination remained open until the late 1980s, when Davenport and Heintz [23] and Weispfenning [81] independently proved double exponential lower bounds. Remarkably, Weispfenning's bound even holds for linear formulas, which contain no products of quantified variables. This result, which was quite disappointing from a practical point of view, triggered another line of research on quantifier elimination and decision procedures of better asymptotic time complexity with respect to particular complexity parameters like numbers of quantifier alternations, quantifiers, and polynomials, or even degrees of polynomials [19, 7, 61, 6, 38]. Being tremendously important in their own right from the theoretical point of view, there are no practical applications or even implementations so far, and there are theoretical arguments for the principal infeasibility of those approaches [41].

A completely different line of research explored virtual substitution-based methods, which originate from the proof of Weispfenning's above-mentioned complexity result. This field was pioneered by Weispfenning and his collaborators [81, 51, 84, 67, 25]. Virtual substitution is an alternative approach that is strong for formulas with low degrees of the quantified variables. Its main advantage is that it is not as sensitive to the number of parameters as CAD. Another advantage of virtual substitution is that it is singly exponential for formulas with a bounded number of quantifier alternations. This is of great practical importance, because mathematical theorems as well as real-world models often lead to real quantifier elimination instances containing rather small numbers of quantifier alternations.

The original description of the virtual substitution method as well as practically all existing improvements and implementations [27, 44] have been focusing on linear and quadratic formulas. The first and probably the best known implementation of the linear and quadratic virtual substitution was done within the computer logic system Redlog [27], which itself is a package of the computer algebra system Reduce. Redlog's original implementation of virtual substitution in practice nicely complements CAD-based algorithms [13, 66, 18] by often coping with quantifier elimination instances that are out of reach for such algorithms. This naturally led to successful applications of virtual substitution-based methods: Redlog was cited more than 350 times in the scientific literature and its linear and quadratic virtual substitution for the reals was successfully applied in various areas of computer science [71], engineering [75, 43], and the sciences [74, 30, 68, 69, 86, 72, 73, 80, 36, 34, 35].

The only notable exception in the "exclusively" linear and quadratic virtual substitution mindset in the literature is, surprisingly, an early work by

Weispfenning [83], where he made precise how to perform virtual substitution up to degree three. Indeed, this approach was quite different from the approaches to linear and quadratic virtual substitution. However, it has never been improved, implemented, or tried in practice.

The motivation and the aim of this dissertation in the context of previous work is to make the next step towards better practical applicability of virtual substitution-based methods by developing novel theoretical as well as practical techniques and software implementations on the grounds of more than two decades of theoretical and practical experiences from linear, quadratic, and cubic virtual substitution. Since virtual substitution can in principle be extended to arbitrary but bounded degrees [84, Section 6], another aim of this dissertation is to show how to do this efficiently, and to lift techniques that are currently applicable only in the linear and quadratic cases to higher degrees.

## 1.1 The Virtual Substitution Method

Any quantifier-free Tarski formula $\varphi$ can be equivalently rewritten as a *positive formula*, which is an $\wedge$-$\vee$-combination of atomic formulas. This is done by using "$\wedge$," "$\vee$," "$\neg$," along with de Morgan's laws to move logical negation inside the scopes of conjunction and disjunction until only atomic formulas occur in its scope, and finally eliminating it altogether by changing the relations of the atomic formulas. For example, an atomic formula $\neg(a > b)$ is replaced with $a \leq b$. Notice that our set of equality and relation symbols $\{=, \neq, <, \leq, \geq, >\}$ of the Tarski language $\mathfrak{L}$ is closed under negation. We will thus restrict ourselves to positive formulas and furthermore we will assume that the right hand side of each atomic formula is zero, which can be easily achieved by subtraction of the right hand side.

Consider $\exists x\big(\varphi(\mathbf{u}, x)\big)$, where $\varphi$ is positive and $\mathbf{u} = u_0, \ldots, u_{m-1}$ are the parameter variables different from $x$. The original idea of virtual substitution is to compute a finite elimination set $E$ for $\varphi$ and $x$ consisting of *elimination terms* $e$ and *substitution guards* $\gamma$ such that

$$\exists x(\varphi) \longleftrightarrow \bigvee_{(\gamma,e)\in E} \gamma \wedge \varphi[x /\!\!/ e]. \tag{1.1}$$

Here $e$ are substituted into the quantifier-free formula $\varphi$ by means of virtual substitution $[x /\!\!/ e]$ applied to every atomic formula in $\varphi$. Virtual substitution $[x /\!\!/ e]$ is a generalization of regular term substitution that maps atomic formulas to quantifier-free formulas.

Let us get an impression what elimination terms and guards look like, how to compute an elimination set, and how to perform virtual substitution. For the sake of simplicity, we restrict ourselves here in the Introduction to formulas containing only relations "$=$," "$\leq$," and "$\geq$."

Without loss of generality, the left hand side term $f$ of an atomic formula $f \ \varrho \ 0$ is a polynomial from $\mathbb{Z}[\mathbf{u}][x]$. Fixing arbitrary real values $\mathbf{a} \in \mathbb{R}^m$ for the parameters $\mathbf{u}$, the satisfying set $\big\{c \in \mathbb{R} \mid \mathbb{R} \models (f \ \varrho \ 0)(\mathbf{a}, c)\big\}$ of an atomic formula $f \ \varrho \ 0$ becomes a finite union of disjoint intervals: The polynomial $f$ is for fixed parameter values $\mathbf{a}$ either identically equal to zero or has finitely many real roots. Since $\varrho \in \{=, \leq, \geq\}$, the intervals constituting the satisfying set of $f \ \varrho \ 0$ for $\mathbf{a}$ are either closed or unbounded from above or below (or from above

and below simultaneously). Obviously, all real endpoints of these intervals are real roots of the polynomial $f$.

Since $\varphi$ is a positive formula, the satisfying set $S = \{c \in \mathbb{R} \mid \mathbb{R} \models \varphi(\mathbf{a}, c)\}$ is obtained from the satisfying sets of atomic formulas occurring in $\varphi$ by union and intersection. Consequently, $S$ is a finite union of disjoint intervals as well, and, again, the endpoints of these intervals are real roots of the left hand sides of the atomic formulas in $\varphi$.

Using this observation we deduce that $\exists x(\varphi)$ holds for $\mathbf{a}$ if and only if there exists a real root $\alpha$ of the left hand side of some atomic formula in $\varphi$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \alpha)$. In other words, the roots of the left hand sides of the atomic formulas in $\varphi$ provide *finitely many* points to try for *any* parameter values $\mathbf{a}$. Intuitively, one replaces in this way an the existential quantifier $\exists x$ ranging over $\mathbb{R}$ with a finite disjunction over the real roots of the left hand sides of atomic formulas in $\varphi$.

Weispfenning's key idea was to use the equivalence of an existential quantifier with a finite disjunction over the roots of the left hand sides to compute a finite elimination set for $\varphi$ and $x$ as follows: Since for any parameter values $\mathbf{a}$ the finitely many roots of the left hand sides of $\varphi$ provide sufficiently many points $\alpha$ to test the validity of $\varphi(\mathbf{a}, \alpha)$, a uniform description of these finitely many points in terms of the parameters $\mathbf{u}$ yields a finite set of, so to say, "uniform parametric numbers" to try. Consequently, testing for which parameter values $\varphi$ holds at some of these "uniform parametric numbers" gives a quantifier-free equivalent of $\exists x(\varphi)$. These two constructs are the underlying concepts of elimination terms and virtual substitution, respectively.

For the linear case, the elimination terms are obtained by symbolically solving for $x$ to obtain a uniform parametric representation of the root. Consider $ax + b \varrho 0$ with $\varrho \in \{=, \leq, \geq\}$ and $a, b \in \mathbb{Z}[\mathbf{u}]$. One takes the formal solution of $ax + b = 0$, i.e., $-\frac{b}{a}$. Since division is obviously not in the Tarski language, one formally constructs the formal solution $-\frac{b}{a}$ in a suitable extension language containing a function symbol for multiplicative inverse. Observe that $-\frac{b}{a}$ is defined over $\mathbb{R}$ only when $a$ is not zero. To this end, the formula $a \neq 0$ is introduced as a guard of the elimination term $-\frac{b}{a}$.

We are now in the situation of Equation (1.1), where we have found $\gamma = (a \neq 0)$ and $e = -\frac{b}{a}$. As an example, for virtual substitution consider $(cx + d \leq 0)\left[x \mathbin{/\!\!/} -\frac{b}{a}\right]$, which yields the quantifier-free Tarski formula $-abc + a^2 d \leq 0$. Notice how in (1.1) that transformation is carried out in conjunction with the guard $\gamma = (a \neq 0)$, which is sufficient for its correctness. Whenever, in contrast $a = 0$, then the corresponding conjunction becomes false. In those cases, the atomic formula $ax + b \varrho 0$ producing our test point does not contain a relevant occurrence of $x$.

For a quadratic atomic formula $ax^2 + bx + c \varrho 0$, where $\varrho \in \{=, \leq, \geq\}$ and $a, b, c \in \mathbb{Z}[\mathbf{u}]$, one takes elimination terms $\frac{-b+\sqrt{\Delta}}{2a}$ and $\frac{-b-\sqrt{\Delta}}{2a}$, where $\Delta$ is the discriminant of $ax^2 + bx + c$. These are terms in a suitable extension language $\mathfrak{L}'$ of $\mathfrak{L}$, which contains function symbols for multiplicative inverse and square root. Both of these elimination terms are defined in $\mathbb{R}$ if and only if $a$ is nonzero and $\Delta$ is non-negative. Consequently, $a \neq 0 \wedge b^2 - 4ac \geq 0$ will be introduced as a guard of both of these test points. The virtual substitution of $\frac{-b\pm\sqrt{\Delta}}{2a}$ into an atomic formula is based on substitution and arithmetic in $\mathfrak{L}'$ combined with

equivalences like

$$a + b\sqrt{c} \leq 0 \longleftrightarrow (a \leq 0 \wedge 0 \leq a^2 - b^2c) \vee (b \leq 0 \wedge a^2 - b^2c \leq 0)$$

that yield a resulting equivalent Tarski formula for virtual substitution into any atomic formula. For further details we refer the reader to [84].

This approach based on extensions of the language $\mathfrak{L}$ using $n$-th root function symbols and multiplicative inverse cannot work beyond degree four because of the famous Abel-Ruffini theorem. The uniform description of finitely many relevant roots in terms of parameters for higher degrees requires a considerably generalized approach. For the cubic case Weispfenning suggested one such possible approach in [83]. The idea there was to look at what a third degree polynomial can look like after fixing parameter values and using the information on the shape of the polynomial to perform virtual substitution using a different type of elimination terms and guards. This thesis picks up on that idea to construct a novel framework for virtual substitution, which can be extended to arbitrary but bounded degree.

Finally, observe that the approach for elimination of $\exists x$ from $\exists x(\varphi)$ can be iterated to eliminate all quantifiers: First, compute a prenex normal form of an input formula. Then proceed by processing the quantifier prefix from the inside to the outside, eliminating one quantifier at a time. In that course, universal quantifiers are translated to existential quantifiers by means of the equivalence $\forall x(\psi) \longleftrightarrow \neg\exists x(\neg\psi)$.

## 1.2 Plan of the Thesis

This Introduction is followed by five core chapters, a concluding chapter, and two appendices.

Chapter 2 is the central chapter of this thesis. We first introduce the notions of parametric root descriptions and candidate solutions, which will be used throughout the whole thesis. Building on these we formally define virtual substitution and show how to perform it. With these tools at hand, we then present the main result of the chapter—a quantifier elimination algorithm scheme. That scheme is subsequently instantiated, by providing three specified subalgorithms, to obtain a standalone virtual substitution-based quantifier elimination algorithm up to degree three of a quantified variable. Then we discuss a new technique called clustering, which has the potential to reduce the size of output formulas obtained by virtual substitution. Finally, we discuss our framework in the context of other existing virtual substitution-based approaches.

In Chapter 3 we extend the framework of Chapter 2 to take the Boolean structure of an input formula into account. We begin by decomposing an input formula into prime constituents, which are subformulas of an input formula that can to some extent be regarded as atomic formulas during quantifier elimination. Then we define the notions of conjunctive associativity and "the DNF" of a Tarski formula. We use these notions to prove a technique called Marking, which underlies the correctness proofs of the structural virtual substitution approach and its variants. Prime constituent decompositions, condensing, and deletion operators are the main blocks of the structural virtual substitution algorithm scheme, which we present and prove correct next. Finally, we propose

novel bound selection strategies based on 0-1 integer linear programming, which exploit the Boolean structure of the input formula as well.

In Chapter 4 we integrate the established technique of degree shift into our virtual substitution framework. We first show how to model a degree shift by virtual substitution. Then we introduce the concept of structural degree shift and show how this can be combined with the ideas from Chapter 3. Furthermore, we discuss connections between structural degree shift and the computation of the DNF.

Chapter 5 focuses on existential Tarski formulas. We discuss the concept of extended quantifier elimination as a method for obtaining in addition to quantifier-free equivalents also parametric sample solutions for the quantified variables. Then we show how to perform extended quantifier elimination by our virtual substitution framework. We also discuss the similarities and differences with respect to extended QE between traditional quadratic QE by virtual substitution and our framework in the context of extended QE. Afterwards we generalize a post-processing method for the quadratic case, which we have introduced in [47], to our new framework. Finally, we report on computational results obtained by our implementation of the post-processing method for the quadratic case from [47].

Chapter 6 reports on our implementation within the computer logic system Redlog and computational experiments carried out with it. Our implementation is standalone up to degree three of a quantified variable, and implements a number of novel concepts developed in this thesis, most notably the structural quantifier elimination algorithm scheme and the clustering technique. It is prepared to be supplemented with other techniques presented in this thesis, e.g., efficient bound selection strategies. We also discuss practical issues one deals with while implementing procedures for elimination of one existential quantifier and of an entire block of existential quantifiers. Furthermore, we present results of comprehensive computational experiments conducted with our implementation.

Chapter 7 summarizes the results obtained throughout this thesis and gives a number of promising future research directions.

Appendix A contains formula schemes used in Chapter 2 to instantiate our QE algorithm scheme for the quadratic and cubic cases. Appendix B contains tables with detailed results and timings from Chapter 6.

## 1.3 Main Contributions

This thesis contributes to both theory and practice of real quantifier elimination by virtual substitution. Our main contributions can be summarized as follows:

1. a novel framework for virtual substitution: We develop a modular algorithm scheme for virtual substitution and precisely specify what needs to be done to obtain complete virtual substitution algorithms via instantiation. Our scheme unifies and improves on the existing approaches for degrees one, two, and three by using our enhancements like smaller elimination sets and clustering. We instantiate the framework for degrees up to three to obtain complete virtual substitution-based QE algorithms. Furthermore, we make clear how to extend the scheme beyond degree three and take first steps into this direction.

2. structural virtual substitution: We propose and prove correct an extension of our virtual substitution framework that takes into account the Boolean structure of an input formula. We show how to compute a prime constituent decomposition, develop structural elimination sets, and structural virtual substitution along with novel bound selection strategies.

3. degree shift transformations: Revisiting an established heuristic to decrease the degree of a quantified variable in an input formula, we reanalyze that transformation within our framework as another instance of virtual substitution. Furthermore, we generalize degree shifts to our structural setting of Chapter 3 and study their potential and limitations.

4. answers and standard answers: We show how to perform extended real QE within our framework. We generalize our answer post-processing approach from [47] for obtaining standard answers for existential problems when the values of the parameters are fixed beforehand.

5. implementation: We present the first complete implementation of a virtual substitution-based QE method beyond degree two. Extensive computational experiments on more than two hundred QE problems that have been considered in the scientific literature over the past twenty years point to a great practical potential of our techniques: Our implementation, which is a part of the computer logic system Redlog, *never* eliminates a lower number of quantifiers than the Redlog's original implementation of linear and quadratic quantifier elimination by virtual substitution. In fact, it often eliminates even more quantifiers.

# Chapter 2

# A Uniform Framework for Virtual Substitution

In the previous chapter we have given an overview of the existing quantifier elimination methods based on virtual substitution, and briefly sketched their key properties. Our discussion there shows that it is not possible to talk about "the" quantifier elimination by virtual substitution, because of the heterogeneity of the existing approaches: The linear [51] and quadratic [84] virtual substitution proceed in a rather algebraic manner, using formal fractions and root expressions to represent the solutions of polynomial constraints. The cubic virtual substitution [83], in contrast, is based on a geometric viewpoint, where roots of a multivariate polynomial are represented as possible shapes of that polynomial after fixing the values of its parameters. In this chapter we introduce and prove correct our general and homogeneous framework unifying all notable virtual substitution-based approaches. The mathematical basis of our framework is the geometric viewpoint of Weispfenning that he used with his cubic virtual substitution algorithm [83].

The main properties of our framework are the following: First, the framework makes no difference between the linear, quadratic, cubic, or even higher degree virtual substitutions. Second, established enhancements like bound selection [85] or factorization will be included in the framework as its inherent parts. Third, we will show how to model all notable virtual substitution-based approaches within our framework. Fourth, the framework can be easily extended beyond degree three by providing three sub-algorithms, which we are going to precisely specify here. Finally, the framework provides a solid basis for an efficient implementation. Our corresponding implementation along with computational experiments will be discussed in Chapter 6.

Recall the principal idea of virtual substitution from Section 1.1. We consider one existential quantifier $\exists x$ at a time. We want to eliminate it from $\exists x(\varphi(\mathbf{u}, x))$, where $\varphi$ a positive formula. During the elimination of $\exists x$, all other variables $\mathbf{u} = (u_0, \ldots, u_{m-1})$ are regarded as parameters. The first step of the actual elimination is essentially done by determining in a uniform way all possible roots of the left hand sides in $\varphi$ in terms of the parameters $\mathbf{u}$. Given such a "uniform parametric number" $r$ and some values $\mathbf{a} \in \mathbb{R}^m$ for the parameters $\mathbf{u}$, $r$ possibly represents a real root. Each "uniform parametric number" $r$ is then

substituted into the whole formula $\varphi$ so that the following holds for any fixed parameter values $\mathbf{a}$: The resulting formula $\varphi[x /\!\!/ r]$ is satisfied by $\mathbf{a}$ if and only if the solution $r$ exists, and $\varphi$ is satisfied by $\mathbf{a}$ and the real number described by the solution $r$ for parameter values $\mathbf{a}$.

Using, e.g., the famous Cardano's formulas, it would be possible to use the ideas of quadratic virtual substitution [84] to obtain virtual substitution algorithms when the degree of $x$ is at most four. However, this was never done explicitly. Even if it was, it would work only for degree three or four, because of the well-known negative results on solvability of polynomial equations with root expressions.

At the same time, it is important to understand that in spite of these negative results, virtual substitution is not at all limited by them. We decided to take the theoretical results of Weispfenning [83] as a basis for our framework. We use his geometric viewpoint and completely avoid the use of formal fractions, root expressions, and higher root symbols. We are going to show how to perform virtual substitution for degree three and that a generalization for arbitrary but bounded degrees is possible.

Our framework not only unifies existing virtual substitution approaches [81, 51, 83, 84, 31], it improves them by employing novel features that are its inherent parts. The following are the most important of these features:

- In the linear case we obtain stronger guards as described in [25, Section 3.6] for granted.
- In comparison with the existing quadratic quantifier elimination, we are able to discard one out of two parametric zeroes of $f$ when considering an atomic formula $f \varrho 0$, where $\varrho \in \{<, >, \leq, \geq\}$. This lifts to degree three and beyond.
- Using our clustering technique, we are able to substitute "more parametric zeroes at once," obtaining shorter quantifier-free equivalents in the cubic case than the existing approach [83].

This chapter is organized as follows: In Section 2.1 we introduce real types and parametric root descriptions. The latter will serve as the fundamental object to represent all roots of a multivariate polynomial w.r.t. parameters. In Section 2.2 we introduce the notion of a candidate solution set, which is in principle a finite set of parametric root descriptions representing all "interesting points" of a satisfying set of a Tarski formula $\varphi(\mathbf{u}, x)$ for any parameter values $\mathbf{a}$. Section 2.3 develops a formal basis for virtual substitution by specifying its semantics and proving its properties. In Section 2.4 we develop a quantifier elimination algorithm scheme based on the notions of previous sections. The scheme is an algorithm parameterized by three precisely specified sub-algorithms. We prove that any quantifier elimination algorithm obtained by instantiating the scheme is correct under the assumption of the correctness of the three sub-algorithms. Then, in Section 2.5 we show how to actually obtain concrete quantifier elimination algorithms by instantiating the algorithm scheme. In Section 2.6 we summarize the properties of the scheme and instantiations thereof and compare them with the existing approaches. Finally, let us note here that the first four sections of this chapter are presented on a rather abstract level, because their main aim is to provide a theoretical basis that will be used throughout the rest of this thesis.

## 2.1 Parametric Root Descriptions

### Univariate Polynomials and Real Types

Let $f \in \mathbb{R}[x]$ be a real univariate polynomial of degree $d$. The *real type* of $f$ is the finite sequence of distinct signs of $f(b)$ as $b \in \mathbb{R}$ traverses the real line from $-\infty$ to $\infty$. If $f$ has no real root, then the sign sequence consists of a single element; it is (1) if and only if $f$ is positive definite and $(-1)$ if and only if $f$ is negative definite. The real type of $f$ is (0) if and only if $f = 0$. If $f$ has $n \geq 1$ distinct real roots, then these roots divide the real line into $2n + 1$ alternating intervals and points over each of which $f$ has constant sign. Hence, the sign sequence has length $2n + 1$ in this case. Accordingly, the real type of $f$ is an alternating sequence consisting of nonzero and zero signs in this case. Real types generally have an odd length.

**Example 1.** For a univariate polynomial $f \in \mathbb{R}[x]$ one can compute all its real roots using established real root isolation methods. With these real roots at hand, it is straightforward to determine the real type of $f$: Evaluate the sign of $f$ for $x$ at $-\infty$, between each pair of successive real roots of $f$, and for $x$ at $\infty$. Evaluation of the sign of $f$ as $x$ tends to $\pm\infty$ can be done using, e.g., the well-known Cauchy upper bound on the magnitude of all roots of $f$.

The real type of $-5x + 3$ is $(1, 0, -1)$. The real type of $x^2 + 1$ is (1), because $x^2 + 1$ is positive definite. The real type of $f = x^3 - 2x^2 - x + 2$ is $(-1, 0, 1, 0, -1, 0, 1)$ because $f = (x + 1)(x - 1)(x - 2)$. The real type of $-x^4 + 4x^3 - 2x^2 - 4x + 3 = (-x - 1)(x - 1)^2(x - 3)$ is $(-1, 0, 1, 0, 1, 0, -1)$. $\diamond$

In the following we will also say that $f$ *realizes* a real type $t$ or that $t$ *is realized* by $f$. In both cases we mean that $f$ has real type $t$.

A sign sequence $t$ is a *real $d$-type* if there exists a univariate polynomial $f \in \mathbb{R}[x]$ of degree exactly $d$ realizing $t$. Using the fundamental theorem of algebra we obtain an upper bound on the number of real $d$-types:

**Proposition 2.** *Let $d$ be a positive integer. Then there exist at most $2^{d+2} - 2$ real $d$-types.*

*Proof.* Consider a real univariate polynomial $f \in \mathbb{R}[x]$ of positive degree $d$, and assume that $f$ has $n \leq d$ distinct real roots. The $n$ distinct real roots divide the real line into $n + 1$ open intervals, each with either positive or negative sign of $f$. This yields at most $2^{n+1}$ possible real types of $f$. Hence, the number of real $d$-types is at most

$$\sum_{n=0}^{d} 2^{n+1} = 2 \cdot (2^{d+1} - 1) = 2^{d+2} - 2. \qquad \square$$

The given upper bound is not tight, because not all $2^{n+1}$ combinations of signs are necessarily realized by a given $d$-degree polynomial. In the following we will show how to compute the exact number of real $d$-types. In contrast to Proposition 2, this computation will not lead to a closed-form expression. We begin with the following lemmas:

**Lemma 3.** *Consider $f \in \mathbb{R}[x]$ of positive degree $d$. Let $m$ be the number of real roots of $f$ counting multiplicities. Then $m \equiv d \mod 2$.*

*Proof.* By the fundamental theorem of algebra there are exactly $d$ complex roots. The non-real roots of a univariate polynomial with real coefficients come in conjugate pairs, which preserves the parity of the number of real roots. □

**Lemma 4.** *Let $(s_1, \ldots, s_{2n+1})$ be a real type of a positive-degree polynomial $f \in \mathbb{R}[x]$ with $n$ distinct real roots $\beta_1 < \cdots < \beta_n$ with multiplicities $m_1, \ldots, m_n$, respectively. Then for each $j \in \{1, \ldots, n\}$ we have $s_{2j+1} = (-1)^{m_j} s_{2j-1}$.*

*Proof.* First observe that $f = (x - \beta_j)^{m_j} r$, where $r \in \mathbb{R}[x]$. Since $f$ has only finitely many real roots, there exists $n \in \mathbb{N} \setminus \{0\}$ such that $f(\beta_j - \frac{1}{n}) \neq 0$, $f(\beta_j + \frac{1}{n}) \neq 0$, and there is exactly one real root, namely $\beta_j$, in the interval $[\beta_j - \frac{1}{n}, \beta_j + \frac{1}{n}]$. Therefore we have:

$$s_{2j-1} = \operatorname{sgn}\Big(f\Big(\beta_j - \frac{1}{n}\Big)\Big) = \operatorname{sgn}\left(\Big(-\frac{1}{n}\Big)^{m_j}\right) \operatorname{sgn}\Big(r\Big(\beta_j - \frac{1}{n}\Big)\Big)$$
$$s_{2j+1} = \operatorname{sgn}\Big(f\Big(\beta_j + \frac{1}{n}\Big)\Big) = \operatorname{sgn}\left(\Big(\frac{1}{n}\Big)^{m_j}\right) \operatorname{sgn}\Big(r\Big(\beta_j + \frac{1}{n}\Big)\Big).$$

The sign of $r$ is constant and nonzero in the interval $[\beta_j - \frac{1}{n}, \beta_j + \frac{1}{n}]$, because $f$ has exactly one real root in that interval. Putting this together with the equations above we finally obtain that $s_{2j+1} = (-1)^{m_j} s_{2j-1}$. □

Lemma 3 and Lemma 4 together imply: A real type $t = (s_1, \ldots, s_{2n+1})$ is realized by some $d$-degree polynomial with a positive leading coefficient if and only if there exists $f \in \mathbb{R}[x]$ with the following properties:

(i) the degree of $f$ is $d$, and the leading coefficient of $f$ is positive,

(ii) $f$ has exactly $n$ distinct real roots $\beta_1, \ldots, \beta_n$, and the multiplicities of these roots are $m_1, \ldots, m_n$, respectively,

(iii) we have $s_1 = (-1)^d$ and $s_{2j+1} = (-1)^{m_j} s_{2j-1}$ for each $j \in \{1, \ldots, n\}$, and

(iv) the following integer constraint system is satisfied:

$$\sum_{j=1}^{n} m_j \equiv d \mod 2, \quad \sum_{j=1}^{n} m_j \leq d. \tag{2.1}$$

The following lemma shows that we can without loss of generality assume that $m_j \in \{1, 2\}$.

**Lemma 5.** *Consider a real type $t = (s_1, \ldots, s_{2n+1})$. There exists a $d$-degree $f \in \mathbb{R}[x]$ with real type $t$ if and only if there exists a $d$-degree $g \in \mathbb{R}[x]$ with real type $t$, and the $n$ distinct real roots $\beta_1 < \cdots < \beta_n$ of $g$ are either simple or double roots of $g$.*

*Proof.* Assume that there exists $f \in \mathbb{R}[x]$, $\deg f = d$, with real type $t$. If the multiplicity of some root $\beta_j$ is greater than two, then we compute $\overline{g} = \frac{f}{(x-\beta_j)^2}(x^2 + 1)$. Observe that the degree of $\overline{g}$ is $d$ and that the real type of $\overline{g}$ is $t$. Applying this step finitely many times we obtain a polynomial $g$ with real type $t$ whose all real roots have multiplicity one or two. The converse implication is obvious. □

Consequently, a particular real type $t$ is realized by $f \in \mathbb{R}[x]$ if and only if the conditions (i)–(iv) above hold for $f$, and $m_j \in \{1, 2\}$ for each $j \in \{1, \ldots, n\}$. Lemma 4 guarantees that $s_1, \ldots, s_{2n+1}$ are uniquely determined by root multiplicities $m_1, \ldots, m_n$ and by the sign of the leading coefficient of $f$, so we obtain: The number $R(d, n)$ of real types realizable by $d$-degree polynomials with a positive leading coefficient and exactly $n$ distinct real zeroes is the number of distinct solutions of the following integer constraint system:

$$\sum_{j=1}^{n} m_j \equiv d \mod 2, \quad \sum_{j=1}^{n} m_j \leq d,$$

where $m_j$ are now *variables* such that $m_j \in \{1, 2\}$. Modeling an assignment to 2 by an assignment to 0 and simplifying yields that this system is equivalent to:

$$\sum_{j=1}^{n} m_j \equiv d \mod 2, \quad \sum_{j=1}^{n} m_j \geq 2n - d, \tag{2.2}$$

where $m_j \in \{0, 1\}$. To sum up, the number $R(d, n)$ of real types realized by $d$-degree polynomials with a positive leading coefficient and exactly $n$ distinct real roots is the number of solutions of (2.2). Since the solutions of (2.2) are simply binary vectors of length $n$ containing at least $2n - d$ nonzero entries, and the parity of the number of the nonzero entries is the same as the parity of $d$, we just directly count the number of binary vectors satisfying these constraints. We obtain that

$$R(d, n) = \sum_{j \geq 0} \binom{n}{(2n - d) + 2j}.$$

To compute the total number of real $d$-types we have to take into account polynomials with a negative leading coefficient. For this we use the following lemma, which follows directly from our definition of real type:

**Lemma 6.** *Let $f \in \mathbb{R}[x]$. Polynomial $f$ is of real type $t = (s_1, \ldots, s_{2n+1})$ if and only if $-f$ is of real type $-t = (-s_1, \ldots, -s_{2n+1})$.* □

Using Lemma 6 and the fact that a $d$-degree polynomial can have at most $d$ distinct real roots we obtain that the total number $T(d)$ of real $d$-types is

$$T(d) = 2 \cdot \sum_{n=0}^{d} R(d, n) = 2 \cdot \sum_{n=0}^{d} \sum_{j \geq 0} \binom{n}{(2n - d) + 2j}. \tag{2.3}$$

It is known [59] that a sum of the form $\sum_{j \geq l} \binom{n}{j}$ cannot be expressed in closed form as a function of $n$ and $l$, unless one resorts to hypergeometric functions. Therefore, it is very unlikely that $T(d)$ can be expressed in a "nice" closed form. Table 2.1 lists the exact numbers $T(d)$ of real $d$-types for degree up to ten. We computed these numbers by evaluating the sum in (2.3).

### Multivariate Polynomials and Parametric Root Descriptions

We now leave the "univariate world" and proceed to multivariate polynomials. Let $f \in \mathbb{Z}[\mathbf{u}][x]$ be a polynomial of positive degree $d$. The $m$ variables $u_0, \ldots, u_{m-1}$ denoted by $\mathbf{u}$ are called the *parameters*. We call $x$ *the main variable*

| $d$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $T(d)$ | 2 | 6 | 8 | 16 | 24 | 42 | 66 | 110 | 176 | 288 |

Table 2.1: Exact numbers of real $d$-types for $d \in \{1, \ldots, 10\}$.

of $\mathbb{Z}[\mathbf{u}][x]$. Throughout the rest of this thesis we use a *recursive representation* that views $f$ as a sum $c_d x^d + \cdots + c_1 x + c_0$, where $c_i \in \mathbb{Z}[\mathbf{u}]$ and $c_d \neq 0$. We define $\operatorname{lc} f = c_d$ and $\operatorname{red} f = c_e x^e + \cdots + c_1 x + c_0$, where $e \in \{0, \ldots, d-1\}$ is maximal such that $c_e \neq 0$. If there is no such $e$, then we define $\operatorname{red} f = 0$. Furthermore, we define $\operatorname{coeffs} f = \{c_0, \ldots, c_d\}$. We denote the first derivative of $f$ with respect to the main variable $x$ by $f'$.

Denote by $\langle \cdot \rangle : \mathbb{Z}[\mathbf{u}] \to \mathbb{R}$ the evaluation homomorphism in postfix notation, i.e., for *parameter values* $\mathbf{a} \in \mathbb{R}^m$ we have $f\langle \mathbf{a} \rangle \in \mathbb{R}[x]$. In case there are no parameters, i.e., $f \in \mathbb{Z}[x]$, we have $f\langle\rangle = f$. A sign sequence $t$ is a *realizable real $d$-type of $f$* if there exist $\mathbf{a} \in \mathbb{R}^m$ such that $f\langle \mathbf{a} \rangle$ has degree $d$ and real type $t$. It is obvious that each realizable real $d$-type of $f$ is also a real $d$-type. The converse, however, is not true, as the following example illustrates:

**Example 7.**
- Realizable real 2-types of $f = x^2 + u_1 x + u_0 \in \mathbb{Z}[u_0, u_1][x]$ are: $(1)$, $(1, 0, 1)$, and $(1, 0, -1, 0, 1)$. The real 2-type $(-1, 0, -1)$, for example, is not realizable by $f$, because the leading coefficient of $f$ is positive.
- Realizable real 2-types of $u_2 x^2 + u_1 x + u_0 \in \mathbb{Z}[u_0, u_1, u_2][x]$ are: $(1)$, $(1, 0, 1)$, $(1, 0, -1, 0, 1)$, $(-1)$, $(-1, 0, -1)$, and $(-1, 0, 1, 0, -1)$. These are actually all the real 2-types.
- Realizable real 1-types of $u_1 x + 5 \in \mathbb{Z}[u_1][x]$ are $(1, 0, -1)$ and $(-1, 0, 1)$. Again, these two real types are all real 1-types. ◇

Let $f \in \mathbb{Z}[\mathbf{u}][x]$ with $d = \deg f > 0$. Let $t = (s_1, \ldots, s_{2n+1})$ be a real $d$-type containing $n \leq d$ zero entries. Let $r \in \{1, \ldots, n\}$ be the number of some root when counting the distinct real roots of $f\langle \mathbf{a} \rangle$ from the left to the right, whereas $\mathbf{a} \in \mathbb{R}^m$ are arbitrary parameter values such that $f\langle \mathbf{a} \rangle$ is of real $d$-type $t$. The number $r$ is called a *root index*, and the pair $(t, r)$ is called a *root specification of $f$*. Let $k \in \mathbb{N} \setminus \{0\}$. Consider a $k$-element set $S = \{(t_1, r_1), \ldots, (t_k, r_k)\}$ of root specifications of $f$ such that $t_i$ are pairwise distinct real $d$-types. The pair $(f, S)$ is called a *parametric root description* of $f$. If $S$ consists of a single pair $(t, r)$, then we will for simplicity write $(f, (t, r))$ instead of $(f, \{(t, r)\})$.

Let $f \in \mathbb{Z}[\mathbf{u}][x]$ with $d = \deg f > 0$. Let $t$ be a real $d$-type. A *guard of real $d$-type $t$ for $f$* is a quantifier-free Tarski formula in the parameters $\mathbf{u}$ that is satisfied by some $\mathbf{a} \in \mathbb{R}^m$ if and only if $f\langle \mathbf{a} \rangle$ is of real $d$-type $t$. Let $k \in \mathbb{N} \setminus \{0\}$. Consider a $k$-element set $S = \{(t_1, r_1), \ldots, (t_k, r_k)\}$ of root specifications of $f$ such that $t_i$ are pairwise distinct real $d$-types. A *guard* of the parametric root description $(f, S)$ is a quantifier-free Tarski formula $\gamma$ in the parameters $\mathbf{u}$ that is satisfied by $\mathbf{a} \in \mathbb{R}^m$ if and only if there exists $j \in \{1, \ldots, k\}$ such that $f\langle \mathbf{a} \rangle$ is of real $d$-type $t_j$. Notice that for any parameter values $\mathbf{a}$ there either exists none or exactly one $t_j$ occurring in $S$ such that $f\langle \mathbf{a} \rangle$ is of real $d$-type $t_j$, because the real types occurring in $S$ are pairwise distinct.

The fact that $\mathbb{R}$ admits quantifier elimination guarantees the existence of a guard of any parametric root description:

**Proposition 8** (Existence of Guards)**.** *Let $f \in \mathbb{Z}[\mathbf{u}][x]$ with $\deg f = d > 0$ and let $k \in \mathbb{N} \setminus \{0\}$. Consider a $k$-element set $S = \big\{(t_1, r_1), \ldots, (t_k, r_k)\big\}$ of root specifications of $f$ such that $t_i$ are pairwise distinct real $d$-types. Then there exists a guard of the parametric root description $(f, S)$.*

*Proof.* Let $j \in \{1, \ldots, k\}$ and let $t_j = (s_1, \ldots, s_{2n+1})$ be such that $n \geq 0$, $s_{2i} = 0$ for each $i \in \{1, \ldots, n\}$, and $s_{2i+1} \in \{-1, 1\}$ for each $i \in \{0, \ldots, n\}$. We define $\gamma'_j$ to be the following Tarski formula:

$$\exists y_2 \ldots \exists y_{2n} \Bigg( \operatorname{lc} f \neq 0 \wedge \bigwedge_{i=1}^{n-1} y_{2i} < y_{2(i+1)} \wedge \bigwedge_{i=1}^{n} f[x \,/\, y_{2i}] = 0 \wedge \\ \forall z \big( z < y_2 \longrightarrow f[x \,/\, z] \; \varrho_1 \; 0 \big) \wedge \forall z \big( z > y_{2n} \longrightarrow f[x \,/\, z] \; \varrho_{2n+1} \; 0 \big) \wedge \\ \bigwedge_{i=1}^{n-1} \forall z \big( y_{2i} < z < y_{2(i+1)} \longrightarrow f[x \,/\, z] \; \varrho_{2i+1} \; 0 \big) \Bigg),$$

where for every $i \in \{0, \ldots, n\}$ we define $\varrho_{2i+1}$ to be "$<$" if $s_{2i+1} = -1$ and $\varrho_{2i+1}$ to be "$>$" if $s_{2i+1} = 1$. The usual substitution of a term $v$ for a variable $x$ into a term $f$ is denoted by $f[x \,/\, v]$.

The formula $\gamma'_j$ asserts that the leading coefficient of $f$ is nonzero, and that there exist exactly $n$ distinct real roots of $f$. Moreover, $\gamma'_j$ also specifies the signs of $f$ at $\pm\infty$ and the signs of $f$ between its $n$ distinct real roots. Therefore, it is not hard to see that $\mathbf{a} \in \mathbb{R}^m$ satisfies $\gamma'_j$ if and only if $f\langle \mathbf{a} \rangle$ has degree $d$ and real type $t_j$, i.e., $\gamma'_j$ is a guard of real $d$-type $t_j$ for $f$. Since $\mathbb{R}$ admits quantifier elimination, there exists a quantifier-free equivalent $\gamma_j$ of $\gamma'_j$. We define $\gamma$ as $\bigvee_{j=1}^{k} \gamma_j$. Then $\mathbf{a} \in \mathbb{R}^m$ satisfies the quantifier-free Tarski formula $\gamma$ in the parameters $\mathbf{u}$ if and only if $f\langle \mathbf{a} \rangle$ has degree $d$ and real type $t_j$ for some $j \in \{1, \ldots, k\}$. □

The purpose of Proposition 8—whose proof is nonconstructive—is merely to show that a guard as we specified it exists. Later we will show how to compute guards of real types and parametric root descriptions for polynomials of degree up to three.

Let $f \in \mathbb{Z}[\mathbf{u}][x]$ with $d = \deg f > 0$, and let $S = \{(t_1, r_1), \ldots, (t_k, r_k)\}$ be a $k$-element set of root specifications such that $t_i$ are pairwise distinct real $d$-types. For the parametric root description $(f, S)$ we define a partial mapping $(f, S)\langle \mathbf{a} \rangle$ from $\mathbb{R}^m$ to $\mathbb{R}$ as follows:

- If $\mathbf{a} \in \mathbb{R}^m$ does not satisfy any guard $\gamma$ of $(f, S)$, then $(f, S)\langle \mathbf{a} \rangle$ is undefined.
- If, in contrast, $\mathbf{a} \in \mathbb{R}^m$ satisfies some guard $\gamma$ of $(f, S)$, then there exist exactly one $j \in \{1, \ldots, k\}$ such that $(t_j, r_j) \in S$, and $f\langle \mathbf{a} \rangle$ is of real $d$-type $t_j$. In this case we define $(f, S)\langle \mathbf{a} \rangle$ to be the $r_j$-th real root of $f\langle \mathbf{a} \rangle$ when counting the distinct real roots of $f\langle \mathbf{a} \rangle$ from the left to the right.

Notice that $(f, S)\langle \mathbf{a} \rangle$ is well-defined when $\mathbf{a}$ satisfies some guard $\gamma$, because the real $d$-types contained in $S$ are pairwise distinct, and for every $(t_j, r_j) \in S$ we defined that $r_j$ is a valid root index for real $d$-type $t_j$.

We say that $(f, S)$ *covers* $\beta \in \mathbb{R}$ *for parameter values* $\mathbf{a}$ if $(f, S)\langle \mathbf{a} \rangle = \beta$. In case there are no parameters, we say that $(f, S)$ covers $\beta \in \mathbb{R}$ if $(f, S)\langle\rangle = \beta$. If

the parameter values $\mathbf{a}$ will be clear from the context, then we will simply say that $(f, S)$ covers $\beta$.

Consider an atomic formula $f \varrho 0$, where $f \in \mathbb{Z}[\mathbf{u}][x]$ and the relation symbol $\varrho$ is one of $\{=, \neq, <, \leq, \geq, >\}$, and let $\mathbf{a} \in \mathbb{R}^m$. The *satisfying set of* $f \varrho 0$ *for parameter values* $\mathbf{a}$ is defined as

$$\Phi(f \varrho 0, \mathbf{a}) = \big\{b \in \mathbb{R} \mid \mathbb{R} \models (f \varrho 0)(\mathbf{a}, b)\big\}.$$

If there are no parameters, i.e., if $f \in \mathbb{Z}[x]$, we shortly write $\Phi(f \varrho 0)$. Since $f\langle\mathbf{a}\rangle$ is a polynomial—and in particular a continuous function—with finitely many roots, the satisfying set $\Phi(f \varrho 0, \mathbf{a})$ is a finite union of disjoint intervals. Furthermore, every endpoint of these intervals is a real root of $f\langle\mathbf{a}\rangle$:

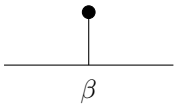
**Proposition 9.** *Let $f \in \mathbb{Z}[\mathbf{u}][x]$ and let $\mathbf{a} \in \mathbb{R}^m$. The satisfying set $\Phi(f \varrho 0, \mathbf{a})$ is a finite union of pairwise disjoint intervals $I_1 \cup \ldots \cup I_l$. Each interval $I_j$ is of one of the following eight forms:*

$$\begin{array}{lll} ]\beta_1, \beta_2[, & [\beta_1, \beta_2], & \beta_1, \\ ]\beta_1, \infty[, & [\beta_1, \infty[, & \\ ]-\infty, \beta_1[, & ]-\infty, \beta_1[, & \\ ]-\infty, \infty[, & & \end{array}$$

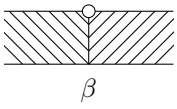
*for some $\beta_1$, $\beta_2 \in \mathbb{R}$. Moreover, we have $f\langle\mathbf{a}\rangle(\beta_1) = f\langle\mathbf{a}\rangle(\beta_2) = 0$.* □

A real number $\beta$ is a *boundary point* of the satisfying set $\Phi(f \varrho 0, \mathbf{a}) = I_1 \cup \ldots \cup I_l$ if it is the infimum or the supremum of some interval $I_j$. Since $f \in \mathbb{Z}[\mathbf{u}][x]$, each boundary point $\beta \in \mathbb{R}$ of $\Phi(f \varrho 0, \mathbf{a})$ is a real root of $f\langle\mathbf{a}\rangle \in \mathbb{R}[x]$. Moreover, $\beta$ is of exactly one of the six types listed below. The satisfying set $\Phi(f \varrho 0, \mathbf{a})$ near $\beta$ can be illustrated as follows:

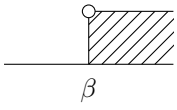
1. isolated point: $\beta \in \Phi(f \varrho 0, \mathbf{a})$, and there exists a positive $\eta \in \mathbb{R}$ such that for all positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\beta - \eta' \notin \Phi(f \varrho 0, \mathbf{a})$ and $\beta + \eta' \notin \Phi(f \varrho 0, \mathbf{a})$,



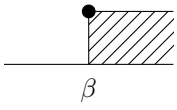

2. excluded point: $\beta \notin \Phi(f \varrho 0, \mathbf{a})$, and there exists a positive $\eta \in \mathbb{R}$ such that for all positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\beta - \eta' \in \Phi(f \varrho 0, \mathbf{a})$ and $\beta + \eta' \in \Phi(f \varrho 0, \mathbf{a})$,



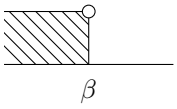

3. strict lower bound: $\beta \notin \Phi(f \varrho 0, \mathbf{a})$, and there exists a positive $\eta \in \mathbb{R}$ such that for all positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\beta - \eta' \notin \Phi(f \varrho 0, \mathbf{a})$ but $\beta + \eta' \in \Phi(f \varrho 0, \mathbf{a})$,



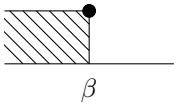

4. weak lower bound: $\beta \in \Phi(f \varrho 0, \mathbf{a})$, and there exists a positive $\eta \in \mathbb{R}$ such that for all positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\beta - \eta' \notin \Phi(f \varrho 0, \mathbf{a})$ but $\beta + \eta' \in \Phi(f \varrho 0, \mathbf{a})$,




5. strict upper bound: defined analogously as the strict lower bound,



6. weak upper bound: defined analogously as the weak lower bound.




The definitions of satisfying sets and boundary points naturally generalize to quantifier-free Tarski formulas $\varphi(\mathbf{u}, x)$. Accordingly, we denote the satisfying set of $\varphi$ for parameter values $\mathbf{a} \in \mathbb{R}^m$ by $\Phi(\varphi, \mathbf{a})$. In the special case of a univariate formula we simply write $\Phi(\varphi)$. Since $\varphi$ is merely a Boolean combination of atomic formulas, and "$\wedge$," "$\vee$," "$\neg$," correspond to intersection, union, and complement on satisfying sets, respectively, we obtain:

**Proposition 10.** *Let $\varphi(\mathbf{u}, x)$ be a Tarski formula and let $\mathbf{a} \in \mathbb{R}^m$. The satisfying set $\Phi(\varphi, \mathbf{a})$ is a finite union of pairwise disjoint intervals $I_1 \cup \ldots \cup I_l$. Each interval $I_j$ is either of one of the eight forms listed in Proposition 9 or of one of the following two additional forms:*

$$]\beta_1, \beta_2], \quad [\beta_1, \beta_2[,$$

*where $\beta_1$, $\beta_2 \in \mathbb{R}$. Moreover, there exist polynomials $f_1$ and $f_2$ occurring as the left hand sides of some atomic formulas in $\varphi$ such that $f_1\langle\mathbf{a}\rangle(\beta_1) = 0$ and $f_2\langle\mathbf{a}\rangle(\beta_2) = 0$.* □

The two new interval forms in Proposition 10 are introduced by the Boolean structure of $\varphi$, as the following example illustrates:

**Example 11.** Consider polynomials $f = x^3 + u_2x^2 + u_0$, $g = x^2 + u_1x + u_0 \in \mathbb{Z}[u_0, u_1, u_2][x]$. Let $\varphi$ be $f < 0 \wedge g \geq 0$.

For parameter values $\mathbf{a} = (1, -4, -5)$ we obtain $f\langle\mathbf{a}\rangle = x^3 - 5x^2 + 1$, $g\langle\mathbf{a}\rangle = x^2 - 4x + 1$, and the satisfying set $\Phi(\varphi, \mathbf{a}) = ]-\infty, \beta_1[ \cup [\beta_2, \beta_3[$, where $\beta_1 \approx -0.42917$ is the first root of $f\langle\mathbf{a}\rangle$, $\beta_2 = 2 + \sqrt{3}$ is the second root of $g\langle\mathbf{a}\rangle$, and $\beta_3 \approx 4.95935$ is the third root of $f\langle\mathbf{a}\rangle$.

For parameter values $\mathbf{a} = (-15, -20, -5)$ we obtain $\Phi(\varphi, \mathbf{a}) = ]-\infty, \beta_1]$, where $\beta_1 = 10 - \sqrt{115}$ is the first root of $g\langle\mathbf{a}\rangle = x^2 - 20x - 15$.

Notice that $\Phi(\varphi, \mathbf{a})$ is unbounded from below for any parameter values $\mathbf{a} \in \mathbb{R}$, because the leading coefficients of both $f$ and $g$ are positive. Since $f$ is cubic and $g$ is quadratic, it follows that there always exists some negative real number satisfying $f < 0 \wedge g \geq 0$. ◇

## 2.2 Candidate Solutions

A *candidate solution* is a triple $(f, S, \tau)$, where $(f, S)$ is a parametric root description of $f \in \mathbb{Z}[\mathbf{u}][x]$ and $\tau$ is one of the tags "IP," "EP," "SLB," "WLB," "SUB," or "WUB." We define $(f, S, \tau)\langle\mathbf{a}\rangle$ as $(f, S)\langle\mathbf{a}\rangle$. Let $\varphi(\mathbf{u}, x)$ be an arbitrary quantifier-free Tarski formula, and let $\mathbf{a}$ be a set $\mathfrak{c}$ of candidate solutions. Then $\mathfrak{c}$ is a *set of candidate solutions for* $\varphi$ if for any parameter values $\mathbf{a} \in \mathbb{R}^m$ the following holds: Every boundary point $\beta \in \mathbb{R}$ of the set $\Phi(\varphi, \mathbf{a})$ is *properly covered by* $\mathfrak{c}$ in the following sense:

(i) If $\beta$ is an isolated point, then there exists a candidate solution $(f, S, \text{IP}) \in \mathfrak{c}$ such that $(f, S)\langle\mathbf{a}\rangle = \beta$, or there exist candidate solutions $(f_1, S_1, \text{WLB})$, $(f_2, S_2, \text{WUB}) \in \mathfrak{c}$ such that $(f_1, S_1)\langle\mathbf{a}\rangle = (f_2, S_2)\langle\mathbf{a}\rangle = \beta$.

(ii) If $\beta$ is an excluded point, then there exists a candidate solution $(f, S, \mathrm{EP}) \in \mathfrak{c}$ such that $(f, S)\langle \mathbf{a} \rangle = \beta$, or there exist candidate solutions $(f_1, S_1, \mathrm{SLB})$, $(f_2, S_2, \mathrm{SUB}) \in \mathfrak{c}$ such that $(f_1, S_1)\langle \mathbf{a} \rangle = (f_2, S_2)\langle \mathbf{a} \rangle = \beta$.

(iii) If $\beta$ is a strict lower bound, then there exists $(f, S, \mathrm{SLB}) \in \mathfrak{c}$ such that $(f, S)\langle \mathbf{a} \rangle = \beta$, or there exist $(f_1, S_1, \mathrm{WLB})$, $(f_2, S_2, \mathrm{EP}) \in \mathfrak{c}$ such that $(f_1, S_1)\langle \mathbf{a} \rangle = (f_2, S_2)\langle \mathbf{a} \rangle = \beta$.

(iv) If $\beta$ is a weak lower bound, then there exists $(f, S, \mathrm{WLB}) \in \mathfrak{c}$ such that $(f, S)\langle \mathbf{a} \rangle = \beta$, or there exist $(f_1, S_1, \mathrm{SLB})$, $(f_2, S_2, \mathrm{IP}) \in \mathfrak{c}$ such that $(f_1, S_1)\langle \mathbf{a} \rangle = (f_2, S_2)\langle \mathbf{a} \rangle = \beta$.

(v) If $\beta$ is a strict upper bound, then the definition is analogous to the case of a strict lower bound.

(vi) If $\beta$ is a weak upper bound, then the definition is analogous to the case of a weak lower bound.

Notice that the notion of "properly covered" demands that any boundary point $\beta$ is covered by candidate solutions with tags that fit the actual type of the boundary point $\beta$: It is *not* sufficient that there exist parametric root descriptions occurring in $\mathfrak{c}$ covering $\beta$. Tags of the covering parametric root descriptions, i.e., candidate solutions, have to be correct as well.

Observe that the syntactic concept of candidate solutions does *not* one-to-one correspond with the semantic concept of boundary points. This is reflected by the "or" clauses in the above definition.

Notice that if the formula $\varphi$ does not contain $x$, then $\emptyset$ is a set of candidate solutions for $\varphi$. Furthermore, any superset of a set of candidate solutions $\mathfrak{c}$ for $\varphi$ is a set of candidate solutions for $\varphi$ as well.

**Example 12.** Consider an atomic formula $f \leq 0$, where $f = ax^2 + 5x - c$ and $a, c \in \mathbb{Z}[\mathbf{u}]$. The following is one possible set of candidate solutions for $f \leq 0$:

1. $(f, ((1, 0, -1, 0, 1), 1), \mathrm{WLB})$,
2. $(f, ((1, 0, -1, 0, 1), 2), \mathrm{WUB})$,
3. $(f, ((1, 0, 1), 1), \mathrm{IP})$,
4. $(f, ((-1, 0, 1, 0, -1), 1), \mathrm{WUB})$,
5. $(f, ((-1, 0, 1, 0, -1), 2), \mathrm{WLB})$,
6. $(5x - c, ((-1, 0, 1), 1), \mathrm{WUB})$.

Observe that the first five candidate solutions properly cover all boundary points of the set $\Phi(f \leq 0, \mathbf{a})$ when $a\langle \mathbf{a} \rangle \neq 0$. For showing this we systematically enumerate all possible real 2-types and analyze what kinds of boundary points the roots of $f$ constitute. Notice that for the real 2-type $(-1, 0, -1)$ there exist no real boundary point, because if $f\langle \mathbf{a} \rangle$ is of real 2-type $(-1, 0, -1)$, then $\Phi(f \leq 0, \mathbf{a}) = ]-\infty, \infty[$

The sixth candidate solution originates from red $f$. It properly covers boundary points of the set $\Phi(f \leq 0, \mathbf{a})$ when $a\langle \mathbf{a} \rangle = 0$. $\diamond$

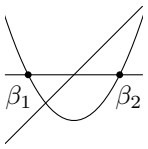



Figure 2.1: A simple picture showing the situation discussed in Example 13.

**Example 13.** Consider the following formula $\varphi(\mathbf{u}, x)$:

$$a - b < 0 \wedge 2x - a - b \geq 0 \wedge x^2 - (a + b)x + ab \geq 0,$$

where $a$, $b \in \mathbb{Z}[\mathbf{u}]$. Even though $\varphi$ looks quite complicated, there exists a set of candidate solutions for $\varphi$ with only one element $(x - b, ((-1, 0, 1), 1), \mathrm{WLB})$. The reason for this is best explained on a simple picture. For any parameter values $\mathbf{a} \in \mathbb{R}^m$ with $\mathbb{R} \models (a - b < 0)\langle\mathbf{a}\rangle$ the polynomials $(2x - a - b)\langle\mathbf{a}\rangle$ and $(x^2-(a+b)x+ab)\langle\mathbf{a}\rangle$—when graphed as functions of the main variable $x$— have the shape shown in Figure 2.1. Furthermore, we have $\beta_1 = a\langle\mathbf{a}\rangle$ and $\beta_2 = b\langle\mathbf{a}\rangle$. From the picture it is obvious that the satisfying set $\Phi(\varphi, \mathbf{a})$ has exactly one weak lower bound, namely $\beta_2 = b\langle\mathbf{a}\rangle$. This is because $\beta_1 = a\langle\mathbf{a}\rangle$ and the root of $(2x-a-b)\langle\mathbf{a}\rangle$—lying between $\beta_1$ and $\beta_2$—are both not relevant for the purpose of parametric root description. ◇

Examples 12 and 13 demonstrate two rather different approaches to compute a set of candidate solutions for a given formula. The approach in Example 12 systematically treats all cases to properly cover all possible boundary points. Example 13 uses in addition geometric information and exploits the structure of the formula to obtain a very simple candidate solution set. Later in this chapter we will show how to systematically compute a set of candidate solutions for $\varphi$ by doing an "exhaustive case analysis," i.e., we will proceed similarly as in Example 12.

The idea with the picture in Example 13, in contrast, actually hides another quantifier elimination problem. Solving that problem we were able to deduce that some roots are generally irrelevant, and compute a simpler candidate solution set. Even though we will not mimic this costly approach in general, in Chapter 3 we will show how a careful analysis of the formula structure can considerably reduce the sizes of the computed candidate solution sets.

Now we continue our exposition and prove some properties of candidate solutions, which will play a central role within the correctness proof of our proposed framework for virtual substitution.

**Proposition 14.** *Let $\varphi_1(\mathbf{u}, x)$ and $\varphi_2(\mathbf{u}, x)$ be two equivalent quantifier-free Tarski formulas. Let $\mathfrak{c}$ be a set of candidate solutions for $\varphi_1$. Then $\mathfrak{c}$ is a set of candidate solutions also for $\varphi_2$.*

*Proof.* Let $\mathbf{a} \in \mathbb{R}^m$ be some values for parameters $\mathbf{u}$. We need to show that any boundary point of the satisfying set $\Phi(\varphi_2, \mathbf{a})$ is properly covered by $\mathfrak{c}$. Let $\beta \in \mathbb{R}$ be a boundary point of $\Phi(\varphi_2, \mathbf{a})$. Since $\varphi_2$ is equivalent to $\varphi_1$, we deduce that $\Phi(\varphi_1, \mathbf{a}) = \Phi(\varphi_2, \mathbf{a})$. Therefore, $\beta$ is a boundary point of $\Phi(\varphi_1, \mathbf{a})$ as well. Finally, the assumption that $\mathfrak{c}$ is a set of candidate solutions for $\varphi_1$ ensures that $\beta$ is properly covered by $\mathfrak{c}$. This proves that $\mathfrak{c}$ is a set of candidate solutions for $\varphi_2$. □

**Proposition 15.** *Let $\varphi_1(\mathbf{u}, x)$ and $\varphi_2(\mathbf{u}, x)$ be quantifier-free Tarski formulas. Let $\mathfrak{c}_1$ and $\mathfrak{c}_2$ be sets of candidate solutions for $\varphi_1$ and $\varphi_2$, respectively. Then $\mathfrak{c}_1 \cup \mathfrak{c}_2$ is a set of candidate solutions for the formulas $\varphi_1 \wedge \varphi_2$ and $\varphi_1 \vee \varphi_2$.*

*Proof.* We prove the proposition only for the formula $\varphi_1 \wedge \varphi_2$. The proof for $\varphi_1 \vee \varphi_2$ is similar.

Let $\mathbf{a} \in \mathbb{R}^m$ be some values for parameters $\mathbf{u}$. We need to prove that any boundary point of the satisfying set $\Phi(\varphi_1 \wedge \varphi_2, \mathbf{a})$ is properly covered by $\mathfrak{c}_1 \cup \mathfrak{c}_2$. To begin with, observe that $\Phi(\varphi_1 \wedge \varphi_2, \mathbf{a}) = \Phi(\varphi_1, \mathbf{a}) \cap \Phi(\varphi_2, \mathbf{a})$. Let $\beta \in \mathbb{R}$ be a boundary point of $\Phi(\varphi_1 \wedge \varphi_2, \mathbf{a})$. We distinguish the following cases:

(a) $\beta$ is an isolated point: Since $\beta \in \Phi(\varphi_1 \wedge \varphi_2, \mathbf{a}) = \Phi(\varphi_1, \mathbf{a}) \cap \Phi(\varphi_2, \mathbf{a})$, we have one of the following two cases:

  (aa) $\beta$ is an isolated point of both sets $\Phi(\varphi_1, \mathbf{a})$ and $\Phi(\varphi_2, \mathbf{a})$, or

  (ab) $\beta$ is w.l.o.g. a weak upper bound of $\Phi(\varphi_1, \mathbf{a})$ and a weak lower bound of $\Phi(\varphi_2, \mathbf{a})$.

  In case (aa), $\beta$ is properly covered by some $(f, S, \mathrm{IP}) \in \mathfrak{c}_1$ or by some $(f_1, S_1, \mathrm{WUB}) \in \mathfrak{c}_1$ and $(f_2, S_2, \mathrm{WLB}) \in \mathfrak{c}_1$. In either case $\beta$ is properly covered by $\mathfrak{c}_1 \cup \mathfrak{c}_2$.

  In case (ab), there are four subcases to consider. However, in all of those subcases we can deduce that $\beta$ is properly covered by $(f, S, \mathrm{IP}) \in \mathfrak{c}_1 \cup \mathfrak{c}_2$ or by $(f_1, S_1, \mathrm{WUB})$, $(f_2, S_2, \mathrm{WLB}) \in \mathfrak{c}_1 \cup \mathfrak{c}_2$. In either case, $\beta$ is properly covered by $\mathfrak{c}_1 \cup \mathfrak{c}_2$.

(b) $\beta$ is a strict lower bound: Here we have one of the following two cases:

  (ba) $\beta$ is a strict lower bound of both sets $\Phi(\varphi_1, \mathbf{a})$ and $\Phi(\varphi_2, \mathbf{a})$, or

  (bb) $\beta$ is w.l.o.g. a weak lower bound of $\Phi(\varphi_1, \mathbf{a})$ and an isolated point of $\Phi(\varphi_2, \mathbf{a})$.

  In case (ba), $\beta$ is properly covered by some $(f, S, \mathrm{SLB}) \in \mathfrak{c}_1$ or by some $(f_1, S_1, \mathrm{WLB}) \in \mathfrak{c}_1$ and $(f_2, S_2, \mathrm{EP}) \in \mathfrak{c}_1$. In either case $\beta$ is properly covered by $\mathfrak{c}_1 \cup \mathfrak{c}_2$.

  In case (bb), there are again four subcases to consider. In all of them we can deduce that $\beta$ is properly covered by $(f, S, \mathrm{SLB}) \in \mathfrak{c}_1 \cup \mathfrak{c}_2$ or by $(f_1, S_1, \mathrm{WLB}) \in \mathfrak{c}_1 \cup \mathfrak{c}_2$ and $(f_2, S_2, \mathrm{EP}) \in \mathfrak{c}_1 \cup \mathfrak{c}_2$.

(c) If $\beta$ is a weak lower bound, a strict upper bound, a weak upper bound, or an excluded point, then we proceed in the same fashion as in cases (a) and (b): Either $\beta$ is the same boundary type of both $\Phi(\varphi_1, \mathbf{a})$ and $\Phi(\varphi_2, \mathbf{a})$, or we need to distinguish four subcases what the situation could exactly look like. In either of the four subcases we can show that $\beta$ is properly covered by $\mathfrak{c}_1 \cup \mathfrak{c}_2$.

We have shown that $\beta$ is properly covered by $\mathfrak{c}_1 \cup \mathfrak{c}_2$ regardless of its boundary type, i.e., the proof of the proposition for $\varphi_1 \wedge \varphi_2$ is finished. □

Proposition 15 does *not* hold for Boolean operator "$\neg$." Consider the atomic formula $f \leq 0$, where $f = ax^2 + 5x - c$ and $a, c \in \mathbb{Z}[\mathbf{u}]$, from Example 12, where we had given a six element candidate solution set $\mathfrak{c}$. Here we show that

$\mathfrak{c}$ is *not* a set of candidate solutions for $\neg(f \leq 0)$, which is obviously equivalent to $f > 0$. Assume that we have parameter values $\mathbf{a}$ such that $a\langle\mathbf{a}\rangle = \frac{25}{4}$ and $c\langle\mathbf{a}\rangle = -1$. Then $f\langle\mathbf{a}\rangle = \frac{1}{4}(5x+2)^2$, so $f\langle\mathbf{a}\rangle$ is of real 2-type $(1,0,1)$. The set $\Phi(f > 0, \mathbf{a})$ has therefore one boundary point, namely the excluded point $-\frac{2}{5}$. This point is not properly covered by $\mathfrak{c}$, because it is only covered by a candidate solution with tag "IP." Since Boolean operators "$\longrightarrow$," and "$\longleftrightarrow$" implicitly use negation, one easily shows that Proposition 15 does not hold for those either.

Notice that we could define a kind of negation of a candidate solution set in the sense that lower bounds would become upper bounds, strict bounds would become weak bounds and vice-versa. An analogy of Proposition 15 would then hold for the remaining Boolean operators (of course when appropriately negating the involved candidate solution sets). However, we do not need this, because we will work only with positive formulas.

Before we continue our exploration of the properties of candidate solution sets, we introduce nonstandard extensions of the real numbers. Recall that $\mathfrak{L}$ denotes the Tarski language, and consider the extension language $\mathfrak{L}_\varepsilon = \mathfrak{L} \cup \{\varepsilon, \infty\}$. By the compactness theorem of first-order logic, there exists a real closed $\mathfrak{L}_\varepsilon$-field $\mathbb{R}^*$ where the interpretation of $\varepsilon$ is a positive infinitesimal with inverse $\infty$, i.e., $\varepsilon$ is greater than zero but smaller than any positive real number, and $\varepsilon \cdot \infty = 1$. The $\mathfrak{L}$-restriction of $\mathbb{R}^*$ is a proper real closed extension field of $\mathbb{R}$. Since real closed fields admit quantifier elimination, they establish in particular a model complete class, so that the extension is an elementary one. Formally, $\mathbb{R}^*|\mathfrak{L} \supset \mathbb{R}$ and $\mathbb{R}^*|\mathfrak{L} \cong \mathbb{R}$. Throughout the rest of this thesis we will denote by $\varepsilon$ a positive infinitesimal with multiplicative inverse $\infty$, and by $\mathbb{R}^*$ we will denote a real closed extension $\mathfrak{L}_\varepsilon$-field we have just described.

**Lemma 16.** *Let $\varphi(\mathbf{u}, x)$ be a Tarski formula. Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values and let $\beta \in \mathbb{R}$. The following statements are equivalent:*

*(i) There exists a positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \beta \pm \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$.*

*(ii) For any positive $\eta \in \mathbb{R}$ there exists positive $\eta' \in \mathbb{R}$ smaller than $\eta$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \beta \pm \eta')$ holds.*

*(iii) $\mathbb{R}^* \models \varphi(\mathbf{a}, \beta \pm \varepsilon)$.*

*Proof.* To prove the lemma we show that (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii) $\Rightarrow$ (i):

- If (i) holds, then (ii) obviously holds as well; just pick for any positive $\eta$ a small $\eta'$ with $\mathbb{R} \models \varphi(\mathbf{a}, \beta \pm \eta')$ whose existence is guaranteed by (i).
- Assume (ii). Assume for a contradiction that $\mathbb{R}^* \not\models \varphi(\mathbf{a}, \beta \pm \varepsilon)$, i.e., $\mathbb{R}^* \models \neg\varphi(\mathbf{a}, \beta \pm \varepsilon)$. It follows that $\mathbb{R}^* \models \neg\varphi[x \,/\, x \pm \varepsilon](\mathbf{a}, \beta)$. Let $n \in \mathbb{N} \setminus \{0\}$. Then we can conclude that $\mathbb{R}^* \models \psi[y \,/\, \varepsilon](\mathbf{a}, \beta)$, where

$$\psi = (0 < y \wedge ny < 1 \wedge \neg\varphi)[x \,/\, x \pm y],$$

  because $\varepsilon$ is a positive infinitesimal, and $\mathbb{R}^* \models (0 < \varepsilon \wedge n\varepsilon < 1)$ holds. Generalizing we obtain that $\mathbb{R}^* \models \exists y \psi(\mathbf{a}, \beta)$. Since $\varepsilon$ does not occur in $\psi$, we restrict from structure $\mathbb{R}^*$ to structure $\mathbb{R}^*|L \cong \mathbb{R}$ and use the elementary equivalence to obtain that $\mathbb{R} \models \exists y \psi(\mathbf{a}, \beta)$. We have just shown

that for any $n \in \mathbb{N} \setminus \{0\}$ there exists $a_0 \in \mathbb{R}$, $0 < a_0 < \frac{1}{n}$, such that $\mathbb{R} \models \neg\varphi(\mathbf{a}, \beta \pm a_0)$. It follows that $\inf S = 0$, where

$$S = \{\, a \in \mathbb{R} \mid 0 < a \text{ and } \mathbb{R} \models \neg\varphi(\mathbf{a}, \beta \pm a) \,\}.$$

At the same time, Proposition 10 guarantees that the satisfying $\Phi(\neg\varphi, \mathbf{a})$ is a finite union of intervals, so the fact that the infimum of $S$ is zero implies that there exists some positive $b_0 \in \mathbb{R}$ such that $\mathbb{R} \models \neg\varphi(\mathbf{a}, \beta \pm b)$ for every positive $b$ smaller than $b_0$. On the other hand, (ii) ensures that there exists a positive $b$ smaller than $b_0$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \beta \pm b)$; which is obviously a contradiction. Thus, (iii) holds.

- Assume that (iii) holds. We prove that (i) holds as well. The key idea here is essentially the same as the idea of the proof of the implication from (ii) to (iii). Since we have $\mathbb{R}^* \models \varphi(\mathbf{a}, \beta \pm \varepsilon)$, we deduce that for any $n \in \mathbb{N} \setminus \{0\}$ there exists $a_0 \in \mathbb{R}$, $0 < a_0 < \frac{1}{n}$, such that $\mathbb{R} \models \varphi(\mathbf{a}, \beta \pm a_0)$. Thus, $\inf S = 0$, where

  $$S = \{\, a \in \mathbb{R} \mid 0 < a \text{ and } \mathbb{R} \models \varphi(\mathbf{a}, \beta \pm a) \,\}.$$

  Proposition 10 guarantees that the satisfying $\Phi(\varphi, \mathbf{a})$ is a finite union of intervals, so the fact that the infimum of $S$ is zero implies that there exists some positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \beta \pm \eta')$ for every positive $\eta'$ smaller than $\eta$. Thus, (i) holds.

We have proven that (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii) $\Rightarrow$ (i), so the proof is finished. □

**Lemma 17.** *Let $\varphi(\mathbf{u}, x)$ be a Tarski formula. Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values. The following statements are equivalent:*

*(i) There exists $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \eta')$ for any $\eta' \in \mathbb{R}$ strictly greater/smaller than $\eta$.*

*(ii) For any $\eta \in \mathbb{R}$ there exists $\eta' \in \mathbb{R}$ strictly greater/smaller than $\eta$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \eta')$ holds.*

*(iii) $\mathbb{R}^* \models \varphi(\mathbf{a}, \pm\infty)$.*

*Proof.* The proof is similar to the proof of Lemma 16, i.e., we show that (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii) $\Rightarrow$ (i):

- If (i) holds, then (ii) obviously holds as well; just pick for any $\eta \in \mathbb{R}$ a real number $\eta'$ greater/smaller than $\eta$ with $\mathbb{R} \models \varphi(\mathbf{a}, \beta \pm \eta')$ whose existence is guaranteed by (i).

- We prove that (ii) implies (iii) for the case $\infty$. The proof for $-\infty$ is similar. Assume that for any $\eta \in \mathbb{R}$ there exists $\eta' \in \mathbb{R}$ greater than $\eta$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \eta')$. We prove that $\mathbb{R}^* \models \varphi(\mathbf{a}, \infty)$. Assume for a contradiction that $\mathbb{R}^* \not\models \varphi(\mathbf{a}, \infty)$, i.e., $\mathbb{R}^* \models \neg\varphi(\mathbf{a}, \infty)$. Let $n \in \mathbb{N}$. Then we can conclude that $\mathbb{R}^* \models \psi[x \,/\, \infty](\mathbf{a})$, where $\psi = (n < x \wedge \neg\varphi)$, because $\infty$ is the multiplicative inverse of a positive infinitesimal $\varepsilon$, and $\mathbb{R}^* \models n < \infty$ holds. Generalizing and using the elementary equivalence of $\mathbb{R}^*$ and $\mathbb{R}$ we obtain that $\mathbb{R} \models \exists x \psi(\mathbf{a})$. We have shown that for any $n \in \mathbb{N}$ we have $\mathbb{R} \models \neg\varphi(\mathbf{a}, n)$. It follows that the set $\Phi(\neg\varphi, \mathbf{a})$ is unbounded from

above. At the same time, Proposition 10 guarantees that the set $\Phi(\neg\varphi, \mathbf{a})$ is a finite union of intervals, so we obtain that there exists $b_0 \in \mathbb{R}$ such that $\mathbb{R} \models \neg\varphi(\mathbf{a}, b)$ for any $b \in \mathbb{R}$ greater than $b_0$. On the other hand, (ii) ensures that there exists $b$ greater than $b_0$ such that $\mathbb{R} \models \varphi(\mathbf{a}, b)$; a contradiction. Thus, (iii) holds.

- The proof of (iii) $\Rightarrow$ (i) is similar to the proof of implication (ii) $\Rightarrow$ (iii), so we omit it.

We have shown that (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii) $\Rightarrow$ (i), so the proof of the lemma is finished. □

**Lemma 18.** *Let $g \in \mathbb{Z}[\mathbf{u}][x]$, and let $\mathbf{a} \in \mathbb{R}^m$ be parameter values such that $g\langle\mathbf{a}\rangle$ is not identically zero. Let $\beta \in \mathbb{R}$. Then the following hold:*

*(i) Let $\varrho \in \{<, >\}$. If $\mathbb{R} \models (g\ \varrho\ 0)(\mathbf{a}, \beta)$, then $\mathbb{R}^* \models (g\ \varrho\ 0)(\mathbf{a}, \beta \pm \varepsilon)$.*

*(ii) $\mathbb{R}^* \models (g \neq 0)(\mathbf{a}, \beta \pm \varepsilon)$.*

*Proof.* (i) We prove this part of the lemma only for "$<$." The proof for "$>$" is similar, so we omit it.

Suppose that $\mathbb{R} \models (g < 0)(\mathbf{a}, \beta)$. Assume for a contradiction that $\mathbb{R}^* \not\models (g < 0)(\mathbf{a}, \beta \pm \varepsilon)$, i.e., $\mathbb{R}^* \models (g \geq 0)(\mathbf{a}, \beta \pm \varepsilon)$. Lemma 16 then ensures that there exists a positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models (g \geq 0)(\mathbf{a}, \beta \pm \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$. Since we assume that $g\langle\mathbf{a}\rangle \neq 0$, it follows that $g\langle\mathbf{a}\rangle \in \mathbb{R}[x]$ has finitely many zeroes. Therefore, there exists a positive $\nu \in \mathbb{R}$ smaller than $\eta$ such that $\mathbb{R} \models (g > 0)(\mathbf{a}, \beta \pm \nu')$ for any positive $\nu' \in \mathbb{R}$ smaller than $\nu$. Recall that we assume $\mathbb{R} \models (g < 0)(\mathbf{a}, \beta)$, i.e., $g\langle\mathbf{a}\rangle$ is negative at $\beta$. This is a contradiction, because we have just shown that continuous function $g\langle\mathbf{a}\rangle$ changes its sign on an interval without hitting the zero. This finishes the proof of (i).

(ii) We again proceed by contradiction. Assume that $\mathbb{R}^* \not\models (g \neq 0)(\mathbf{a}, \beta \pm \varepsilon)$, i.e., $\mathbb{R}^* \models (g = 0)(\mathbf{a}, \beta \pm \varepsilon)$. Lemma 16 then guarantees that there exists a positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models (g = 0)(\mathbf{a}, \beta \pm \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$. This is a contradiction, because we assume that $g\langle\mathbf{a}\rangle \neq 0$, i.e., $g\langle\mathbf{a}\rangle$ has at most finitely many zeroes. □

Now we are ready to prove a crucial property of candidate solution sets, which can be intuitively formulated as follows: If $\mathfrak{c}$ is a set of candidate solutions for $\varphi$, then for any choice of parameter values $\mathbf{a}$ for which $\Phi(\varphi, \mathbf{a}) \neq \emptyset$ it is guaranteed that $\mathfrak{c}$ describes at least one "interesting point $\beta$" such that $(\mathbf{a}, \beta)$ satisfies $\varphi$.

**Theorem 19.** *Let $\varphi(\mathbf{u}, x)$ be a quantifier-free Tarski formula. Let $\mathfrak{c}$ be a set of candidate solutions for $\varphi$. Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values. We define two finite subsets of $\mathbb{R}^*$ as follows:*

$$\begin{aligned}\mathcal{L} = {} & \big\{(f, S)\langle\mathbf{a}\rangle \in \mathbb{R} \mid (f, S, \tau) \in \mathfrak{c} \text{ and } \tau \text{ is "IP" or "WLB"}\big\} \cup \\ & \big\{(f, S)\langle\mathbf{a}\rangle + \varepsilon \in \mathbb{R}^* \mid (f, S, \tau) \in \mathfrak{c} \text{ and } \tau \text{ is "EP" or "SLB"}\big\} \cup \{-\infty\},\end{aligned}$$

$$\begin{aligned}\mathcal{L}' = {} & \big\{(f, S)\langle\mathbf{a}\rangle \in \mathbb{R} \mid (f, S, \tau) \in \mathfrak{c} \text{ and } \tau \text{ is "IP" or "WLB"}\big\} \cup \\ & \big\{(f, S)\langle\mathbf{a}\rangle + \varepsilon \in \mathbb{R}^* \mid (f, S, \tau) \in \mathfrak{c} \text{ and } \tau \text{ is "WLB" or "SLB"}\big\} \cup \{-\infty\}.\end{aligned}$$

*If $\Phi(\varphi, \mathbf{a}) \neq \emptyset$, then the following hold:*

*(i) There exists $\xi \in \mathcal{L}$ such that $\mathbb{R}^* \models \varphi(\mathbf{a}, \xi)$.*

*(ii) There exists $\xi \in \mathcal{L}'$ such that $\mathbb{R}^* \models \varphi(\mathbf{a}, \xi)$.*

*Proof.* Assume that $\Phi(\varphi, \mathbf{a})$ is nonempty, so $\Phi(\varphi, \mathbf{a})$ is the union of pairwise disjoint intervals $I_1, \ldots, I_l$. We prove both parts of the theorem:

(i) Let $I$ be a maximal connected satisfying subset. To begin with, observe that if $I$ is unbounded from below, then Lemma 17 ensures that $\mathbb{R}^* \models \varphi(\mathbf{a}, -\infty)$. Since we obviously have $-\infty \in \mathcal{L}$, (i) follows.

If $I$ is bounded from below, then denote by $\beta \in \mathbb{R}$ its infimum, and distinguish the following two cases:

(a) If $\beta \in I$, then $\beta$ is either an isolated point or a weak lower bound of the satisfying set $\Phi(\varphi, \mathbf{a})$. Since $\mathfrak{c}$ is a set of candidate solutions for $\varphi$, this guarantees that there exists $(f, S, \tau) \in \mathfrak{c}$ such that $\tau$ is "IP" or "WLB" and $\beta = (f, S, \tau)\langle \mathbf{a} \rangle$. In either case $\beta \in \mathcal{L}$. By the definition of the satisfying set, we also have $\mathbb{R}^* \models \varphi(\mathbf{a}, \beta)$.

(b) If $\beta \notin I$, then $\beta$ is either an excluded point or a strict lower bound of the satisfying set $\Phi(\varphi, \mathbf{a})$. Since $\mathfrak{c}$ is a set of candidate solutions for $\varphi$, this implies that there exists $(f, S, \tau) \in \mathfrak{c}$ such that $\tau$ is "EP" or "SLB" and $\beta = (f, S, \tau)\langle \mathbf{a} \rangle$. In either case we have $\beta + \varepsilon \in \mathcal{L}$. By the definition of the satisfying set, we now know that there exists positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \beta + \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$. Lemma 16 now ensures that $\mathbb{R}^* \models \varphi(\mathbf{a}, \beta + \varepsilon)$.

This shows that (i) actually holds for any maximal connected satisfying subset of $\Phi(\varphi, \mathbf{a})$, so the proof of part (i) is finished.

(ii) To begin with, observe that if the satisfying set $\Phi(\varphi, \mathbf{a})$ is unbounded from below, then Lemma 17 ensures that $\mathbb{R}^* \models \varphi(\mathbf{a}, -\infty)$. Since we obviously have $-\infty \in \mathcal{L}'$, (ii) follows.

If $\Phi(\varphi, \mathbf{a})$ is bounded from below, then denote by $\beta \in \mathbb{R}$ its infimum, and distinguish the following two cases:

(a) If $\beta \in \Phi(\varphi, \mathbf{a})$, then the same arguments as given in the proof of part (i) of the theorem show that there exists $\beta \in \mathcal{L}'$ such that $\mathbb{R}^* \models \varphi(\mathbf{a}, \beta)$.

(b) If $\beta \notin \Phi(\varphi, \mathbf{a})$, then $\beta$ is either an excluded point or a strict lower bound of $\Phi(\varphi, \mathbf{a})$.

If $\beta$ was an excluded point of $\Phi(\varphi, \mathbf{a})$, then there would exist a real number $\beta' \in \Phi(\varphi, \mathbf{a})$ such that $\beta' < \beta$; a contradiction with the assumption that $\beta$ is the infimum of $\Phi(\varphi, \mathbf{a})$.

Therefore, $\beta$ is a strict lower bound of $\Phi(\varphi, \mathbf{a})$, and since $\mathfrak{c}$ is a set of candidate solutions for $\varphi$, there exists $(f, S, \mathrm{SLB}) \in \mathfrak{c}$ properly covering $\beta$, or there exist $(f_1, S_1, \mathrm{WLB}) \in \mathfrak{c}$ and $(f_2, S_2, \mathrm{EP}) \in \mathfrak{c}$ properly covering $\beta$. In both cases we have $\beta + \varepsilon \in \mathcal{L}'$. On the other hand, the definition of the satisfying set ensures that there exists positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \beta + \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$. Lemma 16 now ensures that $\mathbb{R}^* \models \varphi(\mathbf{a}, \beta + \varepsilon)$.

In both cases we have just shown that there exists some $\xi \in \mathcal{L}'$ such that $\mathbb{R}^* \models \varphi(\mathbf{a}, \xi)$, so the proof of part (ii) is finished. □

Note that the set $\mathcal{L}$ is obtained by taking those parametric root descriptions that describe lower bounds, and adjusting strict lower bounds by $\varepsilon$. The set $\mathcal{L}'$ is obtained by taking lower bounds while ignoring all exception points, and adjusting both strict and weak lower bounds by $\varepsilon$.

It is straightforward to adjust the proof of Theorem 19 to show that it holds also for the following two finite subsets of $\mathbb{R}^*$:

$$\begin{aligned}\mathcal{U} = \quad & \big\{(f,S)\langle\mathbf{a}\rangle \in \mathbb{R} \mid (f,S,\tau) \in \mathfrak{c} \text{ and } \tau \text{ is “IP” or “WUB”}\big\} \quad \cup \\ & \big\{(f,S)\langle\mathbf{a}\rangle - \varepsilon \in \mathbb{R}^* \mid (f,S,\tau) \in \mathfrak{c} \text{ and } \tau \text{ is “EP” or “SUB”}\big\} \quad \cup \{\infty\},\\ \mathcal{U}' = \quad & \big\{(f,S)\langle\mathbf{a}\rangle \in \mathbb{R} \mid (f,S,\tau) \in \mathfrak{c} \text{ and } \tau \text{ is “IP” or “WUB”}\big\} \quad \cup \\ & \big\{(f,S)\langle\mathbf{a}\rangle - \varepsilon \in \mathbb{R}^* \mid (f,S,\tau) \in \mathfrak{c} \text{ and } \tau \text{ is “WUB” or “SUB”}\big\} \cup \{\infty\}.\end{aligned}$$

Similarly as with $\mathcal{L}$ and $\mathcal{L}'$, the set $\mathcal{U}$ is obtained by taking those parametric root descriptions that describe upper bounds, and adjusting strict upper bounds by $-\varepsilon$. The set $\mathcal{U}'$ is obtained by taking upper bounds while ignoring all exception points, and adjusting both strict and weak upper bounds by $-\varepsilon$.

As already mentioned in the proof of Theorem 19, statement (i) of the theorem actually holds for any maximal connected satisfying subset $I$ of $\Phi(\varphi, \mathbf{a})$ when $\mathcal{L}$ is taken. A careful look at the proof of (ii) reveals that this is *not* the case when $\mathcal{L}'$ is taken, because there could possibly exist an interval $I_j$, $j \in \{1, \ldots, l\}$, whose infimum is an excluded point of the satisfying set $\Phi(\varphi, \mathbf{a})$ that is covered exclusively by a candidate solution with tag “EP.” The following simple example shows such a situation:

**Example 20.** Consider the atomic formula $x - 5 \neq 0$. The candidate solution $(x-5, ((-1,0,1),1), \mathrm{EP})$ constitutes a set of candidate solutions for $x - 5 \neq 0$, because $\Phi(x-5 \neq 0) = ]-\infty, 5[ \cup ]5, \infty[$. Part (ii) of Theorem 19, however, does not hold for maximal connected satisfying subset $]5, \infty[$. The reason is that the infimum 5 of this interval is an excluded point of the satisfying set, which is covered only by a candidate solution with tag “EP.” ◇

### 2.2.1 Candidate Solutions and Factorization

So far we have analyzed the properties of parametric root descriptions and candidate solutions. We did so for arbitrary quantifier-free Tarski formulas. Here, in contrast, we restrict ourselves to atomic formulas of the form $f \varrho 0$, where $f \in \mathbb{Z}[\mathbf{u}][x]$ and the relation $\varrho$ is one of $\{=, \neq, <, \leq, \geq, >\}$. We investigate the special case when we can factorize $f$ into $c \cdot f_1 \cdots f_k$, where $c \in \mathbb{Z}[\mathbf{u}]$ and $f_i \in \mathbb{Z}[\mathbf{u}][x]$ for all $i \in \{1, \ldots, k\}$.

Since $f \varrho 0$ translates to a set of sign conditions involving $f_i$, it is quite natural to expect that it is sufficient to consider any sets of candidate solutions for atomic formulas $f_i \varrho 0$, $i \in \{1, \ldots, k\}$, to derive a set of candidate solutions for $f \varrho 0$. We make things precise and show how to derive a set of candidate solutions for $f \varrho 0$ from any sets of candidate solutions for atomic formulas involving the polynomials $f_i$. We begin our exposition with the following key proposition:

**Proposition 21.** *Let $f \in \mathbb{Z}[\mathbf{u}][x]$ with $\deg f > 0$. Let $f = c \cdot f_1 \cdots f_k$ be a factorization of $f$ such that $c \in \mathbb{Z}[\mathbf{u}]$, $f_i \in \mathbb{Z}[\mathbf{u}][x]$, and $\deg f_i > 0$ for all $i \in K = \{1, \ldots, k\}$. Denote by $\mathfrak{c}_i^{\varsigma}$, where $\varsigma$ is one of $\{=, \neq, <, \leq, \geq, >\}$, a set of candidate solutions for the atomic formula $f_i \varsigma 0$. Then the following hold:*

*(i) If $\varrho \in \{=, \neq\}$, then the set $\mathfrak{c} = \bigcup_{i \in K} \mathfrak{c}_i^{\varrho}$ is a set of candidate solutions for the atomic formula $f \varrho 0$.*

*(ii) If $\varrho \in \{<, >\}$, then the set $\mathfrak{c} = \bigcup_{i \in K} (\mathfrak{c}_i^{<} \cup \mathfrak{c}_i^{>})$ is a set of candidate solutions for the atomic formula $f \varrho 0$.*

*(iii) If $\varrho \in \{\leq, \geq\}$, then the set $\mathfrak{c} = \bigcup_{i \in K} (\mathfrak{c}_i^{\leq} \cup \mathfrak{c}_i^{\geq})$ is a set of candidate solutions for the atomic formula $f \varrho 0$.*

*Proof.* Let $\mathbf{a} \in \mathbb{R}^m$ and consider a boundary point $\beta \in \mathbb{R}$ of the satisfying set $\Phi(f \ \varrho \ 0, \mathbf{a})$. In all three parts of the proposition we have to prove that $\beta$ is properly covered by $\mathfrak{c}$.

To begin with, notice that the fact that $\beta$ is a boundary point of the set $\Phi(f \ \varrho \ 0, \mathbf{a})$ ensures that $f\langle \mathbf{a} \rangle$ is not identically zero, i.e., $c\langle \mathbf{a} \rangle \in \mathbb{R} \setminus \{0\}$ and $f_i\langle \mathbf{a} \rangle \in \mathbb{R}[x] \setminus \{0\}$ for every $i \in K$. Recall that Proposition 9 guarantees that each boundary point of the set $\Phi(f \ \varrho \ 0, \mathbf{a})$ is a root of $f\langle \mathbf{a} \rangle$, so in particular $f\langle \mathbf{a} \rangle(\beta) = 0$. Since $f = c \cdot f_1 \cdots f_k$ and $c\langle \mathbf{a} \rangle \in \mathbb{R} \setminus \{0\}$, this implies that $f_i\langle \mathbf{a} \rangle(\beta) = 0$ for at least one $i \in K$.

Now we are ready to prove the three parts of the proposition:

(i) We show (i) for the case when $\varrho$ is "$=$." The proof for the case when $\varrho$ is "$\neq$" is similar. Since we consider the atomic formula $f = 0$ and $f\langle \mathbf{a} \rangle$ is not identically zero, $\beta$ has to be an isolated point such that $f_i\langle \mathbf{a} \rangle(\beta) = 0$ for at least one $i \in K$. Recall that $f_i\langle \mathbf{a} \rangle \in \mathbb{R}[x] \setminus \{0\}$, so $\beta$ is a boundary point of the set $\Phi(f_i = 0, \mathbf{a})$ as well. On the other hand, we assume that $\mathfrak{c}_i^{=}$ is a set of candidate solutions for $f_i = 0$, so there exists either $(f, S, \mathrm{IP}) \in \mathfrak{c}_i^{=} \subseteq \mathfrak{c}$ such that $(f, S)$ covers $\beta$, or there exist $(f_1, S_1, \mathrm{WLB})$, $(f_2, S_2, \mathrm{WUB}) \in \mathfrak{c}_i^{=} \subseteq \mathfrak{c}$ such that both $(f_1, S_1)$ and $(f_2, S_2)$ cover $\beta$. In either case we deduce that $\beta$ is properly covered by $\mathfrak{c}$. This finishes the proof of (i).

(ii) We show (ii) for the case when $\varrho$ is "$<$." The proof for the case when $\varrho$ is "$>$" is similar. Let $I = \{i \in K \mid f_i\langle \mathbf{a} \rangle(\beta) = 0\}$. Since $f = c \cdot f_1 \cdots f_k$ and $c\langle \mathbf{a} \rangle \in \mathbb{R} \setminus \{0\}$, $f_i\langle \mathbf{a} \rangle \in \mathbb{R}[x] \setminus \{0\}$ for every $i \in K$, we deduce that $I \neq \emptyset$. In the following we denote by $\varepsilon$ a positive infinitesimal. The definition of $I$ together with Lemma 18 ensure that the following two equations hold:

$$\operatorname{sgn}\big(f\langle \mathbf{a} \rangle(\beta - \varepsilon)\big) = \operatorname{sgn}\big(c\langle \mathbf{a} \rangle(\beta - \varepsilon)\big) \cdot \prod_{i \in K} \operatorname{sgn}\big(f_i\langle \mathbf{a} \rangle(\beta - \varepsilon)\big) =$$
$$\operatorname{sgn}\big(c\langle \mathbf{a} \rangle\big) \cdot \prod_{i \in I} \operatorname{sgn}\big(f_i\langle \mathbf{a} \rangle(\beta - \varepsilon)\big) \cdot \prod_{i \in K \setminus I} \operatorname{sgn}\big(f_i\langle \mathbf{a} \rangle(\beta)\big)$$

and

$$\operatorname{sgn}\big(f\langle\mathbf{a}\rangle(\beta+\varepsilon)\big) = \operatorname{sgn}\big(c\langle\mathbf{a}\rangle(\beta+\varepsilon)\big)\cdot\prod_{i\in K}\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon)\big) = \operatorname{sgn}\big(c\langle\mathbf{a}\rangle\big)\cdot\prod_{i\in I}\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon)\big)\cdot\prod_{i\in K\setminus I}\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta)\big).$$

All sign expressions in both of these equations are nonzero, because both $\operatorname{sgn}\big(f\langle\mathbf{a}\rangle(\beta-\varepsilon)\big)$ and $\operatorname{sgn}\big(f\langle\mathbf{a}\rangle(\beta+\varepsilon)\big)$ are nonzero. Isolating in both equations the common term $\operatorname{sgn}(c\langle\mathbf{a}\rangle)$, and putting these together we obtain that

$$\frac{\operatorname{sgn}\big(f\langle\mathbf{a}\rangle(\beta-\varepsilon)\big)}{\prod_{i\in I}\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta-\varepsilon)\big)} = \frac{\operatorname{sgn}\big(f\langle\mathbf{a}\rangle(\beta+\varepsilon)\big)}{\prod_{i\in I}\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon)\big)}. \tag{2.4}$$

Since we consider the atomic formula $f < 0$ and $f\langle\mathbf{a}\rangle$ is not identically zero, $\beta$ has to be a strict lower bound, a strict upper bound, or an exception point of the set $\Phi(f < 0, \mathbf{a})$. We distinguish these three cases:

1. $\beta$ is a strict lower bound: In this case we have $\operatorname{sgn}(f\langle\mathbf{a}\rangle(\beta-\varepsilon)) = 1$ and $\operatorname{sgn}(f\langle\mathbf{a}\rangle(\beta+\varepsilon)) = -1$: Plugging these values into Equation (2.4) yields
$$-\prod_{i\in I}\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta-\varepsilon)\big) = \prod_{i\in I}\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon)\big).$$
Thus, $\operatorname{sgn}(f_i\langle\mathbf{a}\rangle(\beta-\varepsilon)) \neq \operatorname{sgn}(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon))$ for at least one $i \in I$. Now there are two cases to consider:
If $\operatorname{sgn}(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon)) = -1$, then $\beta$ is a strict lower bound of the set $\Phi(f_i < 0, \mathbf{a})$. Since $\mathfrak{c}_i^<$ is a set of candidate solutions for $f_i < 0$, $\mathfrak{c}_i^< \subseteq \mathfrak{c}$ properly covers $\beta$.
If $\operatorname{sgn}(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon)) = 1$, then $\beta$ is a strict lower bound of the set $\Phi(f_i > 0, \mathbf{a})$. Since $\mathfrak{c}_i^>$ is a set of candidate solutions for $f_i > 0$, $\mathfrak{c}_i^> \subseteq \mathfrak{c}$ properly covers $\beta$.
We have proven that $\beta$ is properly covered by $\mathfrak{c}$ in both cases, so the proof of (ii) for the case when $\beta$ is a strict lower bound is finished.
2. $\beta$ is a strict upper bound: Now we have $\operatorname{sgn}(f\langle\mathbf{a}\rangle(\beta-\varepsilon)) = -1$ and $\operatorname{sgn}(f\langle\mathbf{a}\rangle(\beta+\varepsilon)) = 1$. Similarly as in the previous case, plugging these values into Equation (2.4) we obtain that there exists at least one $i \in I$ such that $\operatorname{sgn}(f_i\langle\mathbf{a}\rangle(\beta-\varepsilon)) \neq \operatorname{sgn}(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon))$. Using similar arguments as in the previous case, we show that $\beta$ is properly covered by $\mathfrak{c}$. This finishes the proof of (ii) for the case when $\beta$ is a strict upper bound.
3. $\beta$ is an exception point: It holds that $\operatorname{sgn}(f\langle\mathbf{a}\rangle(\beta-\varepsilon)) = -1$ and $\operatorname{sgn}(f\langle\mathbf{a}\rangle(\beta+\varepsilon)) = -1$: Plugging these values into Equation (2.4) we obtain
$$\prod_{i\in I}\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta-\varepsilon)\big) = \prod_{i\in I}\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon)\big).$$
This implies that
   (a) there exists at least one $i \in I$ such that
$$\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta-\varepsilon)\big) = \operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon)\big), \quad \text{or}$$

(b) there exist $i, j \in I$, where $i \neq j \in I$ such that

$$\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta-\varepsilon)\big) = \operatorname{sgn}\big(f_j\langle\mathbf{a}\rangle(\beta+\varepsilon)\big) = 1,$$
$$\operatorname{sgn}\big(f_i\langle\mathbf{a}\rangle(\beta+\varepsilon)\big) = \operatorname{sgn}\big(f_j\langle\mathbf{a}\rangle(\beta-\varepsilon)\big) = -1.$$

In case (a), $\beta$ is either an exception point of the set $\Phi(f_i < 0, \mathbf{a})$, or $\beta$ is an exception point of the set $\Phi(f_i > 0, \mathbf{a})$. Therefore, $\mathfrak{c} = \mathfrak{c}_i^< \cup \mathfrak{c}_i^>$ properly covers $\beta$.

In case (b), $\beta$ is a strict lower bound of the set $\Phi(f_i < 0, \mathbf{a})$ and a strict upper bound of the set $\Phi(f_j < 0, \mathbf{a})$. Therefore, there exists $(f, S, \mathrm{EP}) \in \mathfrak{c}_i^< \cup \mathfrak{c}_j^>$ such that $(f, S)$ covers $\beta$, or there exist $(f_1, S_1, \mathrm{SLB})$ and $(f_2, S_2, \mathrm{SUB})$ from $\mathfrak{c}_i^< \cup \mathfrak{c}_j^>$ such that both $(f_1, S_1)$ and $(f_2, S_2)$ cover $\beta$. This means that $\beta$ is properly covered by $\mathfrak{c} = \mathfrak{c}_i^< \cup \mathfrak{c}_j^>$.

(iii) The proof is similar to the proof of part (ii), so we omit it. □

Let us at this point state a few well-known equivalences that hold in the theory of the real closed fields:

**Proposition 22.** *The following equivalences hold in* $\mathbb{R}$ *for any* $f$, $g \in \mathbb{Z}[\mathbf{u}][x]$ *and any positive integer* $m$*:*

$$\begin{aligned}
f \cdot g = 0 &\longleftrightarrow f = 0 \vee g = 0,\\
f \cdot g \neq 0 &\longleftrightarrow f \neq 0 \wedge g \neq 0,\\
f \cdot g > 0 &\longleftrightarrow f > 0 \wedge g > 0 \vee f < 0 \wedge g < 0,\\
f \cdot g < 0 &\longleftrightarrow f < 0 \wedge g > 0 \vee f > 0 \wedge g < 0,\\
g^{2m} = 0 &\longleftrightarrow g = 0,\\
g^{2m} \neq 0 &\longleftrightarrow g \neq 0,\\
g^{2m} < 0 &\longleftrightarrow \text{false},\\
g^{2m} \leq 0 &\longleftrightarrow g = 0,\\
g^{2m} \geq 0 &\longleftrightarrow \text{true},\\
g^{2m} > 0 &\longleftrightarrow g \neq 0,\\
g^{2m+1}\ \varrho\ 0 &\longleftrightarrow g\ \varrho\ 0, \quad \textit{where } \varrho \in \{=, \neq, <, \leq, \geq, >\}. \quad \square
\end{aligned}$$

With these equivalences at hand, we sketch an alternative proof of Proposition 21: Consider an atomic formula of the form $c \cdot f_1 \cdots f_k = 0$. The first equivalence of Proposition 22 ensures that this atomic formula is equivalent to $c = 0 \vee f_1 = 0 \vee \cdots \vee f_k = 0$. Now we repeatedly apply Proposition 15 to deduce that the union of candidate solutions for formulas $f_i = 0$ yields a set of candidate solutions for $c = 0 \vee f_1 = 0 \vee \cdots \vee f_k = 0$. Proposition 14 then ensures that this set is also a set of candidate solutions for the formula $c \cdot f_1 \cdots f_k = 0$. This proves part (i) of Proposition 21 for the case when $\varrho$ is "=."

Consider now an atomic formula of the form $c \cdot f_1 \cdots f_k < 0$. Applying repeatedly the third and the fourth equivalence of Proposition 22 we end up with a formula of length $O(2^k)$ containing only atomic formulas of the form $c \gtrless 0$ and $f_i \gtrless 0$. Repeatedly applying Proposition 15 then ensures that the union of candidate solutions for formulas $f_i \gtrless 0$ yields a set of candidate solutions for

formula $c \cdot f_1 \cdots f_k < 0$. This proves part (ii) of Proposition 21 for the case when $\varrho$ is "$<$."

Observe that we do not need to write down the equivalent formula of length $O(2^k)$ explicitly to construct a set of candidate solutions for $c \cdot f_1 \cdots f_k < 0$. In fact, the formula $c \cdot f_1 \cdots f_k < 0$ hides the Boolean complexity behind the algebraic complexity exhibited by a polynomial of higher degree. This way of "hiding" the Boolean complexity behind the algebraic complexity has an interesting consequence: A set of candidate solutions for $c \cdot f_1 \cdots f_k < 0$ obtained by Proposition 21, i.e., by considering the factors of $f$, possibly contains redundant candidate solutions. The actual reason for this will become clear in Chapter 3 in the context of conjunctive associativity, where we will analyze the Boolean structure of such a formula. Here we illustrate this redundancy on a simple example:

**Example 23.** Consider the atomic formula $f_1 \cdot f_2 < 0$, where $f_1 = x^2 - 47 \in \mathbb{R}[x]$ and $f_2 = -x + 5 \in \mathbb{R}[x]$. Sets of candidate solutions for atomic formulas $f_1 \gtrless 0$ and $f_2 \gtrless 0$ are listed in Table 2.2. Note that these sets are minimal, i.e., no candidate solution can be removed from them.

Proposition 21 guarantees that the union of the sets listed in Table 2.2 is a set of candidate solutions for atomic formula $f_1 \cdot f_2 < 0$. This is indeed the case, because the satisfying set $\Phi(f_1 \cdot f_2 < 0)$ is $]-\sqrt{47}, 5[ \cup ]\sqrt{47}, \infty[$, and $(f_1, ((1,0,-1,0,1),1), \mathrm{SLB})$ covers $-\sqrt{47}$, $(f_1, ((1,0,-1,0,1),2), \mathrm{SLB})$ covers $\sqrt{47}$, and $(f_2, ((1,0,-1),1), \mathrm{SUB})$ covers 5. Therefore, all boundary points are properly covered. At the same time, we see that the candidate solution $(f_2, ((1,0,-1),1), \mathrm{SLB})$ is redundant, because it covers the boundary point 5, which is not a strict lower bound. Similarly, $(f_1, ((1,0,-1,0,1),1), \mathrm{SUB})$ covers $-\sqrt{47}$, and $(f_1, ((1,0,-1,0,1),2)\}, \mathrm{SUB})$ covers $\sqrt{47}$, but neither $-\sqrt{47}$ nor $\sqrt{47}$ is a strict upper bound of the satisfying set $\Phi(f_1 \cdot f_2 < 0)$. It would be actually sufficient to take

$$\begin{aligned}\big\{&(f_1, ((1,0,-1,0,1),1), \mathrm{SLB}),\\ &(f_1, ((1,0,-1,0,1),2), \mathrm{SLB}),\\ &(f_2, ((1,0,-1),1), \mathrm{SUB})\big\}\end{aligned}$$

to obtain a set of candidate solutions for $f_1 \cdot f_2 < 0$.

Our example also shows that Proposition 21 cannot be strengthened in the sense that we need to include *both* sets of candidate solutions generated by "$<$" as well as by "$>$:" In our example, taking the union of the sets of candidate solutions only for $f_1 < 0$ and $f_2 < 0$ *does not* yield a set of candidate solutions for $f_1 \cdot f_2 < 0$, because the strict lower bound $\sqrt{47}$ is not properly covered, and the strict upper bound 5 is not properly covered. Similarly, taking the union of the sets of candidate solutions for $f_1 > 0$ and $f_2 > 0$ *does not* yield a set of candidate solutions for $f_1 \cdot f_2 < 0$, because the strict lower bound $-\sqrt{47}$ is not properly covered. ◇

## 2.3 Semantics of Virtual Substitution

Let $f \in \mathbb{Z}[\mathbf{u}][x]$ be a polynomial of positive degree $d$. Let $(f, S)$, where $S = \{(t_1, r_1), \ldots, (t_k, r_k)\}$, be a parametric root description of $f$. A *virtual substitu-*

| atomic formula | set of candidate solutions |
|---|---|
| $f_1 < 0$ | $\big\{(f_1, ((1,0,-1,0,1),1), \text{SLB}),$ $(f_1, ((1,0,-1,0,1),2), \text{SUB})\big\}$ |
| $f_1 > 0$ | $\big\{(f_1, ((1,0,-1,0,1),2), \text{SLB}),$ $(f_1, ((1,0,-1,0,1),1), \text{SUB})\big\}$ |
| $f_2 < 0$ | $\big\{(f_2, ((1,0,-1),1), \text{SLB})\big\}$ |
| $f_2 > 0$ | $\big\{(f_2, ((1,0,-1),1), \text{SUB})\big\}$ |

Table 2.2: Sets of candidate solutions for atomic formulas in Example 23.

*tion* $[x /\!\!/ (f,S)]$ of $(f,S)$ for $x$ is a mapping from atomic formulas to quantifier-free formulas. We denote the quantifier-free formula obtained by virtual substitution applied to an atomic formula $g \varrho 0$ by $(g \varrho 0)[x /\!\!/ (f,S)]$. Semantically, $(g \varrho 0)[x /\!\!/ (f,S)]$ is a formula $F$ in the parameters $\mathbf{u}$ such that for any parameter values $\mathbf{a} \in \mathbb{R}^m$ the following implication holds: If $\mathbf{a}$ satisfies a guard $\gamma$ of $(f,S)$, then $\mathbf{a}$ satisfies $F$ if and only if $\mathbb{R} \models (g \varrho 0)(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle)$. Notice that the assumption that $\mathbf{a}$ satisfies $\gamma$ ensures that $(f,S)\langle\mathbf{a}\rangle$ is defined and yields a real number.

**Proposition 24** (Existence of Virtual Substitution)**.** *Let $(f,S)$, where $S = \{(t_1,r_1),\ldots,(t_k,r_k)\}$, be a parametric root description of $f \in \mathbb{Z}[\mathbf{u}][x]$. Let $g \varrho 0$ be an atomic formula. Then there exists a quantifier-free Tarski formula $F$ in the parameters $\mathbf{u}$ meeting our specification of virtual substitution.*

*Proof.* Let $j \in \{1,\ldots,k\}$ and let $t_j = (s_1,\ldots,s_{2n+1})$ be such that $n \geq 0$, $s_{2i} = 0$ for each $i \in \{1,\ldots,n\}$, and $s_{2i+1} \in \{-1,1\}$ for each $i \in \{0,\ldots,n\}$. Let $\gamma_j$ be a guard of the real type $t_j$ for $f$. We define $F'_j$ to be the following Tarski formula:

$$\gamma_j \wedge \exists y_1 \ldots \exists y_n \left( \bigwedge_{i=1}^{n-1} y_i < y_{i+1} \wedge \bigwedge_{i=1}^{n} f[x \,/\, y_i] = 0 \wedge (g \varrho 0)[x \,/\, y_{r_j}] \right).$$

Since $\mathbb{R}$ admits quantifier elimination, there exists a quantifier-free equivalent $F_j$ of $F'_j$. We define $F$ as $\bigvee_{j=1}^{k} F_j$.

Let now $\mathbf{a} \in \mathbb{R}^m$ be some parameter values satisfying a guard $\gamma$ of $(f,S)$. Since the real types $\{t_1,\ldots,t_k\}$ occurring in $S$ are pairwise distinct, there exists exactly one $t_j$ such that $f\langle\mathbf{a}\rangle$ is of real type $t_j = (s_1,\ldots,s_{2n+1})$. Consequently, the guard $\gamma_j$ of the real type $t_j$ for $f$ is satisfied by $\mathbf{a}$ and the guards $\gamma_i$, $i \neq j$ of the real types $t_i$ for $f$ are not satisfied. This means that $f\langle\mathbf{a}\rangle$ has exactly $n$ distinct real roots $\beta_1, \ldots, \beta_n$, and $(f,S)\langle\mathbf{a}\rangle = \beta_{r_j}$ because $(t_j,r_j) \in S$. Therefore, $F'_j$ is satisfied if and only if $y_i$ are assigned the real numbers $\beta_i$, respectively, such that $\mathbb{R} \models (g \varrho 0)(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle)$ holds. □

Proposition 24 is similar to Proposition 8, which stated the existence of a guard for a parametric root description. The purpose of Proposition 24 is also similar: It merely states that a virtual substitution formula with the specified semantics exists. To actually compute a virtual substitution formula we could use any quantifier elimination algorithm for the reals and eliminate the quantifiers from the formulas $F'_j$ in the proof of Proposition 24.

Computing quantifier-free equivalents of the formulas $F'_j$ when $f = a_n x^n + \cdots + a_0$ and $g = b_m x^m + \cdots + b_0$, where $a_i$ and $b_j$ are parameters, would yield a finite number of "offline" formula schemes that could be used to perform virtual substitution as we have defined it at the beginning of this section when the degrees of "producing" and "target" polynomials are not higher than $n$ and $m$, respectively. However, in practice this turned out to be infeasible by methods like CAD already for $n = 3$ and $m = 1$ because of a high number of the parameters. In Section 2.5 we will develop a self-contained approach that derives "offline" virtual substitution formula schemes and guards without any external quantifier elimination algorithm. We will make explicit virtual substitution formulas for the case when $n \leq 3$ and show how our approach generalizes to arbitrary but bounded degree of $f$.

The term substitution of a rational $\frac{a}{b}$, $a \in \mathbb{Z}$, $b \in \mathbb{N} \setminus \{0\}$, for $x$ into an atomic formula $g \ \varrho \ 0$, where $g = c_d x^d + \cdots + x_1 x + c_0 \in \mathbb{Z}[\mathbf{u}][x]$, $c_i \in \mathbb{Z}[\mathbf{u}]$, and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$, yields the following atomic formula in a suitable extension language $\mathfrak{L}'$ of $\mathfrak{L}$:

$$c_d\Big(\frac{a}{b}\Big)^d + \cdots + c_1\Big(\frac{a}{b}\Big) + c_0 \ \varrho \ 0.$$

Equivalently rewriting the result as an $\mathfrak{L}$-formula we obtain:

$$c_d a^d b^0 + \cdots + c_1 a b^{d-1} + c_0 a^0 b^d \ \varrho \ 0.$$

In the following we overload the notation for the $\mathfrak{L}'$-term substitution, and denote this atomic $\mathfrak{L}$-formula by $(g \ \varrho \ 0)[x \,/\, \frac{a}{b}]$. According to our definition of virtual substitution, this atomic $\mathfrak{L}$-formula is equivalent to the formula $(g \ \varrho \ 0)\big[x /\!\!/ \big(bx - a, ((-1, 0, 1), 1)\big)\big]$ for the following reasons:

(i) $bx - a \in \mathbb{Z}[x]$ is of real type $(-1, 0, 1)$ regardless of parameter values $\mathbf{a}$,

(ii) a guard of the parametric root description $\big(bx - a, ((-1, 0, 1), 1)\big)$ is therefore "true," and

(iii) $\big(bx - a, ((-1, 0, 1), 1)\big)\langle \mathbf{a} \rangle = \frac{a}{b}$, because the first and only real root of $bx - a$ is obviously $\frac{a}{b}$ for any parameter values $\mathbf{a}$.

This shows that

$$\mathbb{R} \models \Big((g \ \varrho \ 0)\big[x /\!\!/ \big(bx - a, ((-1, 0, 1), 1)\big)\big] \longleftrightarrow (g \ \varrho \ 0)\Big[x \,/\, \frac{a}{b}\Big]\Big)(\mathbf{a})$$

for arbitrary parameter values $\mathbf{a} \in \mathbb{R}^m$. Observe that the $\mathfrak{L}'$-term substitution $[x \,/\, t]$ of an arbitrary $\mathfrak{L}'$-term $t$ for $x$ is modeled by virtual substitution in a similar fashion as the $\mathfrak{L}'$-term substitution of a rational $[x \,/\, \frac{a}{b}]$ we have just described.

The mapping $[x /\!\!/ (f, S)]$ naturally generalizes to quantifier-free Tarski formulas $\varphi$, by applying $[x /\!\!/ (f, S)]$ to each atomic formula occurring in $\varphi$. This is compatible with our definition of virtual substitution:

**Theorem 25** (Semantics of Virtual Substitution)**.** *Let $\varphi(\mathbf{u}, x)$ be a quantifier-free Tarski formula. Let $(f, S)$, where $f \in \mathbb{Z}[\mathbf{u}][x]$, be a parametric root description with a guard $\gamma(\mathbf{u})$. Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values satisfying $\gamma$. Then $\mathbb{R} \models \varphi[x /\!\!/ (f, S)](\mathbf{a})$ if and only if $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle \mathbf{a} \rangle)$.*

*Proof.* Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values satisfying $\gamma$. Therefore, $(f, S)\langle\mathbf{a}\rangle$ is defined and yields a real number. We proceed by structural induction on the Boolean structure of $\varphi$. To begin with, observe that if $\varphi$ is an atomic formula, then the theorem directly follows from our definition of virtual substitution $[x /\!\!/ (f, S)]$.

Assume that $\varphi$ is of the form $\varphi_1 \wedge \varphi_2$, and that the theorem holds for $\varphi_1$ and $\varphi_2$. Since $\varphi[x /\!\!/ (f, S)]$ is obtained by applying $[x /\!\!/ (f, S)]$ to each atomic formula of $\varphi$, we have $\varphi[x /\!\!/ (f, S)] = \varphi_1[x /\!\!/ (f, S)] \wedge \varphi_2[x /\!\!/ (f, S)]$. By the induction hypothesis, $\mathbb{R} \models \varphi_i[x /\!\!/ (f, S)](\mathbf{a})$ if and only if $\mathbb{R} \models \varphi_i(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle)$ for $i \in \{1, 2\}$. Therefore,

$$\mathbb{R} \models (\varphi_1 \wedge \varphi_2)[x /\!\!/ (f, S)](\mathbf{a}) \quad \text{if and only if} \quad \mathbb{R} \models (\varphi_1 \wedge \varphi_2)(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle),$$

which proves the statement of the theorem for the case when $\varphi = \varphi_1 \wedge \varphi_2$.

Assume now that $\varphi$ is of the form $\neg\varphi'$, and that the theorem holds for $\varphi'$. We have $\varphi[x /\!\!/ (f, S)] = \neg(\varphi'[x /\!\!/ (f, S)])$, and by the induction hypothesis $\mathbb{R} \not\models \varphi'[x /\!\!/ (f, S)](\mathbf{a})$ if and only if $\mathbb{R} \not\models \varphi'(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle)$. Thus,

$$\mathbb{R} \models (\neg\varphi')[x /\!\!/ (f, S)](\mathbf{a}) \quad \text{if and only if} \quad \mathbb{R} \models (\neg\varphi')(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle).$$

The proof is similar for other Boolean operators, so we omit it. □

At this point we would like to explain a rather subtle fact: If we allowed root specifications with duplicate real types to occur in the set $S$ of root specifications of a parametric root description $(f, S)$, then Theorem 25 would *not* hold. Another natural consequence is that $(f, S)\langle\mathbf{a}\rangle$ would then be a mapping from $\mathbb{R}^m$ into finite subsets of $\mathbb{R}$. We illustrate this on a simple example.

Consider $f = x^2 - 4x + 3 \in \mathbb{Z}[\mathbf{u}][x]$. This polynomial has two real roots (namely 1 and 3), and it is of real type $t = (1, 0, -1, 0, 1)$. Consider set $S = \{(t, 1), (t, 2)\}$ of root specifications, which obviously contains duplicate real types. Since $f$ contains no parameters, the parametric root description $(f, S)$ has guard "true," and specifies both real roots of $f$ at once, i.e., $(f, S)\langle\mathbf{a}\rangle$ would yield $\{1, 3\}$ for any parameter values $\mathbf{a}$. Now we consider two formulas:

- $(x - 2 < 0 \wedge x - 2 > 0)[x /\!\!/ (f, S)]$: Semantically, this formula should be obviously "false," because neither the first nor the second real root of $f$ is both smaller and greater than 2.

- $(x-2 < 0)[x /\!\!/ (f, S)] \wedge (x-2 > 0)[x /\!\!/ (f, S)]$: According to our definition of virtual substitution, both formulas evaluate to "true," because there exists a real root of $f$ specified by $(t, 1) \in S$—namely 1—that is smaller than 2. Similarly, there exists a real root of $f$ that is specified by $(t, 2) \in S$—namely 3—that is greater than 2. Therefore, the whole formula evaluates to "true."

This is definitely not what we expect from a reasonable definition of virtual substitution. Therefore, the constraint in our definition of parametric root description does not allow the set of root specifications to contain duplicate real types, so we cannot substitute more than one real root per real type, and Theorem 25 holds. Finally, notice that there exists a virtual substitution of parametric root description $(f, S)$ from our example, but this virtual substitution is not compatible with the Boolean connective "$\wedge$," as we have just shown.

### 2.3.1 Virtual Substitution and Pseudo Remaindering

Our definition of virtual substitution is nonconstructive, and we postponed its realization to later sections. At the same time, Proposition 24 guarantees the existence of an algorithm with the following specification:

Specification of Algorithm `vs-prd-at`$\big(g \varrho 0, (f, S), x\big)$:
Input: an atomic formula $g \varrho 0$ and a parametric root description $(f, S)$ such that $g$, $f \in \mathbb{Z}[\mathbf{u}][x]$, $\varrho \in \{=, \neq, <, \leq, \geq, >\}$, and $\deg g < \deg f$.
Output: $(g \varrho 0)[x /\!\!/ (f, S)]$, i.e., a quantifier-free formula in the parameters $\mathbf{u}$ meeting our specification of virtual substitution.

Regardless of the actual realization of algorithm `vs-prd-at`, we introduce here a variant of pseudo remaindering that will allow us to extend the applicability of `vs-prd-at` to target atomic formulas $g \varrho 0$, where $g$ is of arbitrarily high degree:

Algorithm `pseudo-sgn-rem`$(g, f, x)$.
Input: polynomials $g, f \in \mathbb{Z}[\mathbf{u}][x]$ such that $\deg f > 0$.
Output: a polynomial $h \in \mathbb{Z}[\mathbf{u}][x]$ with the following properties:

(i) $\deg h < \deg f$

(ii) For any parameter values $\mathbf{a} \in \mathbb{R}^m$ such that $(\operatorname{lc} f)\langle \mathbf{a} \rangle \neq 0$ and for any value $\beta \in \mathbb{R}$ of $x$ the following implication holds: If $f\langle \mathbf{a} \rangle(\beta) = 0$, then $\operatorname{sgn}\big(h\langle \mathbf{a} \rangle(\beta)\big) = \operatorname{sgn}\big(g\langle \mathbf{a} \rangle(\beta)\big)$.

1. While $\deg g \geq \deg f$ do

   1.1. If $\operatorname{lc} f$ divides $\operatorname{lc} g$, then

   $$g := \operatorname{red} g - \frac{\operatorname{lc} g}{\operatorname{lc} f} x^{\deg g - \deg f} \operatorname{red} f.$$

   1.2. If $\operatorname{lc} f$ is positive semi-definite, then

   $$g := \operatorname{lc} f \operatorname{red} g - (\operatorname{lc} g) x^{\deg g - \deg f} \operatorname{red} f.$$

   1.3. If $\operatorname{lc} f$ is negative semi-definite, then

   $$g := -\operatorname{lc} f \operatorname{red} g + (\operatorname{lc} g) x^{\deg g - \deg f} \operatorname{red} f.$$

   1.4. If none of the conditions above holds, then

   $$g := (\operatorname{lc} f)^2 \operatorname{red} g - (\operatorname{lc} f)(\operatorname{lc} g) x^{\deg g - \deg f} \operatorname{red} f.$$

2. Return $g$.

**Lemma 26.** *Algorithm* `pseudo-sgn-rem` *meets its specification.*

*Proof.* To begin with, observe that if $\deg g < \deg f$, then the lemma holds, because $h = g$. Therefore, we assume that $\deg g \geq \deg f$. It is not hard to check that the degree of $g$ lowers with each iteration of the while loop, so the algorithm terminates and returns a polynomial of strictly lower degree than $f$.

Let us now analyze one iteration of the while loop. Denote the value of $g$ before the iteration by $g_1$. Denote the value of $g$ after the iteration by $g_2$. Assume that property (ii) holds for $g_1$. We show that property (ii) holds for $g_2$ as well.

Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values such that $(\operatorname{lc} f)\langle\mathbf{a}\rangle \neq 0$ and let $\beta \in \mathbb{R}$ be such that $f\langle\mathbf{a}\rangle(\beta) = 0$. We need to show that $\operatorname{sgn}\big(g_2\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g_1\langle\mathbf{a}\rangle(\beta)\big)$. To achieve this, we first show that for polynomial

$$r = (\operatorname{lc} f)^2 \operatorname{red} g_1 - (\operatorname{lc} f)(\operatorname{lc} g_1) x^{\deg g_1 - \deg f} \operatorname{red} f$$

we have $\operatorname{sgn}\big(r\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g_1\langle\mathbf{a}\rangle(\beta)\big)$. Using the obvious fact that for any $p \in \mathbb{Z}[\mathbf{u}][x]$ we have $p = (\operatorname{lc} p)x^{\deg p} + \operatorname{red} p$, we obtain the following:

$$\begin{aligned}
&(\operatorname{lc} f)^2 g_1 - (\operatorname{lc} f)(\operatorname{lc} g_1) x^{\deg g_1 - \deg f} f = \\
&\quad = (\operatorname{lc} f)^2 \big((\operatorname{lc} g_1) x^{\deg g_1} + \operatorname{red} g_1\big) \\
&\qquad\qquad - (\operatorname{lc} f)(\operatorname{lc} g_1) x^{\deg g_1 - \deg f} \big((\operatorname{lc} f) x^{\deg f} + \operatorname{red} f\big) = \\
&\quad = (\operatorname{lc} f)^2 (\operatorname{lc} g_1) x^{\deg g_1} + (\operatorname{lc} f)^2 \operatorname{red} g_1 \\
&\qquad\qquad - (\operatorname{lc} f)^2 (\operatorname{lc} g_1) x^{\deg g_1} - (\operatorname{lc} f)(\operatorname{lc} g_1) x^{\deg g_1 - \deg f} \operatorname{red} f = \\
&\quad = (\operatorname{lc} f)^2 \operatorname{red} g_1 - (\operatorname{lc} f)(\operatorname{lc} g_1) x^{\deg g_1 - \deg f} \operatorname{red} f = \\
&\quad = r.
\end{aligned}$$

Recall that we assume $f\langle\mathbf{a}\rangle(\beta) = 0$. Using this together with the identity $r = (\operatorname{lc} f)^2 g_1 - (\operatorname{lc} f)(\operatorname{lc} g_1) x^{\deg g_1 - \deg f} f$ shows that $r\langle\mathbf{a}\rangle(\beta) = \big((\operatorname{lc} f)^2 g_1\big)\langle\mathbf{a}\rangle(\beta)$. On the other hand, we also assume $(\operatorname{lc} f)\langle\mathbf{a}\rangle \neq 0$, so $\operatorname{sgn}\big((\operatorname{lc} f)^2\langle\mathbf{a}\rangle\big) = 1$. Thus

$$\operatorname{sgn}\big(r\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g_1\langle\mathbf{a}\rangle(\beta)\big),$$

and we distinguish four cases, which correspond to the four cases in the body of the while loop:

1. $\operatorname{lc} f$ divides $\operatorname{lc} g$: In this case we obtain $g_2 = \frac{r}{(\operatorname{lc} f)^2}$. Since $\operatorname{sgn}\big((\operatorname{lc} f)^2\langle\mathbf{a}\rangle\big) = 1$, this implies that $\operatorname{sgn}\big(g_2\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g_1\langle\mathbf{a}\rangle(\beta)\big)$.

2. $\operatorname{lc} f$ is positive semi-definite: Here we have $g_2 = \frac{r}{\operatorname{lc} f}$ and $\operatorname{sgn}\big((\operatorname{lc} f)\langle\mathbf{a}\rangle\big) = 1$, because we assume that $\operatorname{sgn}\big((\operatorname{lc} f)\langle\mathbf{a}\rangle\big) \neq 0$. Thus, we conclude that $\operatorname{sgn}\big(g_2\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g_1\langle\mathbf{a}\rangle(\beta)\big)$ in this case as well.

3. $\operatorname{lc} f$ is negative semi-definite: Now we have $g_2 = \frac{r}{-\operatorname{lc} f}$, and it follows that $\operatorname{sgn}\big((-\operatorname{lc} f)\langle\mathbf{a}\rangle\big) = 1$, because we assume $\operatorname{sgn}\big((\operatorname{lc} f)\langle\mathbf{a}\rangle\big) \neq 0$. Again, we obtain $\operatorname{sgn}\big(g_2\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g_1\langle\mathbf{a}\rangle(\beta)\big)$.

4. none of the above: Here we have $g_2 = r$, so $\operatorname{sgn}\big(g_2\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g_1\langle\mathbf{a}\rangle(\beta)\big)$ follows.

We showed that in each of the four cases we have $\operatorname{sgn}\big(g_1\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g_2\langle\mathbf{a}\rangle(\beta)\big)$.

Finally, by induction on the number of iterations of the while loop, we obtain that $\operatorname{sgn}\big(h\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g\langle\mathbf{a}\rangle(\beta)\big)$, i.e., the property (ii) holds for $h$ returned by the algorithm. □

Lemma 26 guarantees that it is correct to use `pseudo-sgn-rem` as a preprocessing that ensures that the degree of $g$ is strictly smaller than the degree of $f$ when calling `vs-prd-at`:

**Proposition 27.** *Let $(f, S)$, where $S = \{(t_1, r_1), \ldots, (t_k, r_k)\}$, be a parametric root description of $f \in \mathbb{Z}[\mathbf{u}][x]$. Let $g \varrho 0$, where $g \in \mathbb{Z}[\mathbf{u}][x]$, be an atomic formula. Denote by $h$ the polynomial returned by* `pseudo-sgn-rem`$(g, f, x)$. *Denote by $\psi$ the formula returned by* `vs-prd-at`$(h \varrho 0, f, x)$. *Then for $\mathbf{a} \in \mathbb{R}^m$ satisfying a guard $\gamma$ of $(f, S)$ we have*

$$\mathbb{R} \models \big((g \varrho 0)[x /\!\!/ (f, S)] \longleftrightarrow \psi\big)(\mathbf{a}).$$

*Proof.* We need to show that $\mathbf{a}$ satisfies $(g \varrho 0)[x /\!\!/ (f, S)]$ if and only if $\mathbf{a}$ satisfies formula $\psi$.

Assume that $\mathbb{R} \models (g \varrho 0)[x /\!\!/ (f, S)](\mathbf{a})$. Since $\mathbf{a}$ satisfies $\gamma$, $f\langle\mathbf{a}\rangle$ is of some real type $t_j$, whereas $(t_j, r_j) \in S$. Using Theorem 25 we obtain that $\mathbb{R} \models (g \varrho 0)(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle)$. At the same time, $\gamma$ is satisfied by $\mathbf{a}$ if and only if $f\langle\mathbf{a}\rangle$ is of some real type $t_j$ occurring in $S$ *and* the degree of $f\langle\mathbf{a}\rangle$ is $\deg f$. This ensures that $f\langle\mathbf{a}\rangle$ is a polynomial of degree $\deg f$, i.e., $(\operatorname{lc} f)\langle\mathbf{a}\rangle \neq 0$. Since $(f, S)\langle\mathbf{a}\rangle$ is a root of $f\langle\mathbf{a}\rangle$, the specification of algorithm `pseudo-sgn-rem` guarantees that $\operatorname{sgn}\big(h\langle\mathbf{a}\rangle(\beta)\big) = \operatorname{sgn}\big(g\langle\mathbf{a}\rangle(\beta)\big)$. Therefore, $\mathbb{R} \models (h \varrho 0)(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle)$. Theorem 25 together with the specification of Algorithm `vs-prd-at` and Lemma 26 now ensure that $\mathbb{R} \models \psi(\mathbf{a})$.

The proof of the converse implication is similar, so we omit it. □

We would like to emphasize that the polynomial $h$ computed by algorithm `pseudo-sgn-rem` is *not* exactly the pseudo remainder of $g$ pseudo divided by $f$. Our algorithm `pseudo-sgn-rem` is more careful about the sign of $\operatorname{lc} f$ to ensure that condition (ii) holds. The following example illustrates the difference:

**Example 28.** Consider polynomials $g = x^2 + u_2 \in \mathbb{Z}[u_0, u_1, u_2][x]$ and $f = u_0x^2 - x + u_1 \in \mathbb{Z}[u_0, u_1, u_2][x]$. The pseudo remainder of $g$ and $f$ is $r = x + u_0u_2 - u_1$, and we have $u_0 g = f + r$. The polynomial $r$, however, does *not* satisfy condition (ii): For $u_0 = -2$, $u_1 = 6$, $u_2 = 1$, $x = -2$ we have $f = 0$ and $g = 5$ but $r = -10$, so in particular $\operatorname{sgn} g \neq \operatorname{sgn} r$ for these values of the parameters and $x$. On the other hand, the polynomial $h = u_0x + u_0^2u_2 - u_0u_1$ computed by `pseudo-sgn-rem` satisfies condition (ii) for any assignment to the parameters $u_0$, $u_1$, $u_2$, and the variable $x$. ◇

### 2.3.2 Virtual Substitution with Nonstandard Symbols

Let us remind ourselves of the key idea of quantifier elimination by virtual substitution: To obtain a necessary and sufficient condition for $\exists x(\varphi)$ to hold we evaluate $\varphi(\mathbf{u}, x)$ at finitely many "interesting" test points. These "interesting" test points have the following property for any parameter values $\mathbf{a} \in \mathbb{R}^m$: If $\Phi(\varphi, \mathbf{a})$ is nonempty, then there exists an "interesting" test point that is in the set $\Phi(\varphi, \mathbf{a})$. This replaces the existential quantification $\exists x$ over the infinite field $\mathbb{R}$ with evaluation of $\varphi$ at a finite set of test points.

One way to obtain an "interesting" set of test points for $\varphi$ is by means of candidate solutions as introduced in Section 2.2. These are parametric root descriptions that represent potential interval endpoints of the satisfying set $\Phi(\varphi, \mathbf{a})$

for any $\mathbf{a}$. The actual evaluation of $\varphi$ at a test point is subsequently realized by virtual substitution of a parametric root description using algorithm `vs-prd-at`.

This approach has the following caveat: If, for some parameter values $\mathbf{a}$, the satisfying set of $\Phi(\varphi, \mathbf{a})$ contains an open interval, for example when we allow relations $\{\neq, <, >\}$ to occur in $\varphi$, we may simply miss that open interval: We evaluate $\varphi$ *exactly* at interval endpoints. This could then lead to a quantifier-free "equivalent" that is not satisfied by $\mathbf{a}$ even though $\Phi(\varphi, \mathbf{a})$ is nonempty. Therefore, the adjustment by $\varepsilon$ and $-\infty$ in Theorem 19 is necessary: We need a concept that would evaluate $\varphi$ infinitesimally close to a test point, so that we can guarantee to hit an open interval as well. For similar reasons, it may be necessary to evaluate $\varphi$ as $x$ tends to $\pm\infty$. This is where the virtual substitution involving nonstandard symbols enter the game. The description of what exactly is sufficient to substitute to guarantee the correctness of the quantifier elimination approach will be made precise in Section 2.4, where we will give a quantifier elimination algorithm scheme. Now we continue with defining semantics of virtual substitutions involving nonstandard symbols.

Formally, we define the virtual substitution of the nonstandard symbol $\infty$ into an atomic formula as follows: $(g\ \varrho\ 0)[x /\!\!/ \infty]$ is a quantifier-free Tarski formula in the parameters $\mathbf{u}$ that is satisfied by parameter values $\mathbf{a} \in \mathbb{R}^m$ if and only there exists $\eta \in \mathbb{R}$ such that $\mathbb{R} \models (g\ \varrho\ 0)(\mathbf{a}, \eta')$ holds for any $\eta' \in \mathbb{R}$ greater than $\eta$. Intuitively, $\mathbf{a}$ satisfies $(g\ \varrho\ 0)[x /\!\!/ \infty]$ if and only if $g\langle\mathbf{a}\rangle$ has "the correct" sign as $x$ tends to $\infty$. The virtual substitution of $-\infty$ is defined analogously.

Algorithm `vs-inf-at` performs the specified virtual substitution of nonstandard symbols $\pm\infty$.

Algorithm `vs-inf-at`$(g\ \varrho\ 0, \iota, x)$.
Input: an atomic formula $g\ \varrho\ 0$, where $g = c_d x^d + \cdots + c_1 x + c_0 \in \mathbb{Z}[\mathbf{u}][x]$, $c_i \in \mathbb{Z}[\mathbf{u}]$, and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$. The nonstandard part $\iota$ is $\pm\infty$.
Output: a quantifier-free Tarski formula $(g\ \varrho\ 0)[x /\!\!/ \iota]$.

1. If $\deg g < 1$, then return $g\ \varrho\ 0$.
2. If $\varrho$ is "=," then return $\bigwedge_{i=0}^{d} c_i = 0$.
3. If $\varrho$ is "$\neq$," then return $\bigvee_{i=0}^{d} c_i \neq 0$.
4. If $\iota = \infty$ or $\deg g$ is even, then $h := \operatorname{lc} g$, else $h := -\operatorname{lc} g$.
5. If $\varrho \in \{<, \leq\}$, then assign "$<$" to $\varsigma$. If $\varrho \in \{\geq, >\}$, then assign "$>$" to $\varsigma$.
6. Return $\big(h\ \varsigma\ 0 \vee (h = 0 \wedge$ `vs-inf-at`$(\operatorname{red} g\ \varrho\ 0, \iota, x))\big)$.

**Proposition 29.** *Algorithm* `vs-inf-at` *meets its specification.*

*Proof.* We prove the proposition only for the case when $\iota$ is $\infty$. The proof for $\iota = -\infty$ is similar, so it is omitted. We proceed by induction on the degree $d$ of $g$. To begin with, observe that when $d = \deg g < 1$, then `vs-inf-at` trivially meets its specification.

In the following we assume that $d > 0$ and that `vs-inf-at` meets its specification when given a polynomial of degree strictly smaller than $d$. Let $\mathbf{a} \in \mathbb{R}^m$ be arbitrary parameter values. We distinguish cases depending on the relation symbol $\varrho$:

- $\varrho$ is "=:" A real univariate polynomial has arbitrarily large real roots if and only if it is identically zero. Therefore, $g\langle\mathbf{a}\rangle \in \mathbb{R}[x]$ is zero for arbitrarily large real numbers if and only if its coefficients simultaneously vanish, or equivalently $g\langle\mathbf{a}\rangle = 0$. This is equivalent to the statement that formula $\bigwedge_{i=0}^{d} c_i = 0$ returned by the algorithm is satisfied by parameter values $\mathbf{a}$. This proves that $(g = 0)[x /\!\!/ \infty]$ is $\bigwedge_{i=0}^{d} c_i = 0$, so the algorithm meets its specification in this case.

- $\varrho$ is "<:" Now observe that real univariate polynomial $g\langle\mathbf{a}\rangle \in \mathbb{R}[x]$ is negative for arbitrarily large values of $x$ if and only if its leading coefficient is negative, or when its leading coefficient is zero but $(\operatorname{red} g < 0)[x /\!\!/ \infty]$ holds. Since $\iota = \infty$ the algorithm sets $h$ to $\operatorname{lc} g$. Therefore, the returned formula is of the form $\operatorname{lc} g < 0 \vee \big(\operatorname{lc} g = 0 \wedge \texttt{vs-inf-at}(\operatorname{red} g \ \varrho \ 0, \infty, x)\big)$, which holds exactly when the leading coefficient is negative or when the leading coefficient vanishes but $(\operatorname{red} g < 0)[x /\!\!/ \infty]$ holds. Since we assume that `vs-inf-at` meets its specification for polynomials of degree strictly smaller than $d$, we obtain that $\texttt{vs-inf-at}(\operatorname{red} g \ \varrho \ 0, \infty, x)$ is $(\operatorname{red} g < 0)[x /\!\!/ \infty]$, so the resulting formula is indeed $(g < 0)[x /\!\!/ \infty]$.

If $\varrho \in \{\neq, \leq, \geq, >\}$, then the proof is done by a similar case distinction on the sign of $\operatorname{lc} g$ as for "=" and "<," so we omit it. □

To illustrate the algorithm on an example, let us consider the virtual substitution $(g \leq 0)[x /\!\!/ -\infty]$ for $g = u_4x^4 + (u_2 - 7)x^2 - 5x + u_0 \in \mathbb{Z}[u_0, \ldots, u_4]$. The output of Algorithm $\texttt{vs-inf-at}(g \leq 0, \infty, x)$ is

$$u_4 < 0 \vee \Big(u_4 = 0 \wedge \big(u_2 - 7 < 0 \vee (u_2 - 7 = 0 \wedge (5 < 0 \vee 5 = 0 \wedge u_0 < 0))\big)\Big).$$

Obviously, this formula is equivalent to

$$u_4 < 0 \vee (u_4 = 0 \wedge u_2 - 7 < 0).$$

Now we continue with virtual substitutions involving $\pm\varepsilon$. The virtual substitution of a parametric root description $(f, S)$ plus a positive infinitesimal $\varepsilon$ into an atomic formula $g \ \varrho \ 0$ is a quantifier-free Tarski formula in the parameters $\mathbf{u}$ denoted by $(g \ \varrho \ 0)[x /\!\!/ (f, S) + \varepsilon]$. Semantically, $(g \ \varrho \ 0)[x /\!\!/ (f, S) + \varepsilon]$ is a formula $F$ such that for parameter values $\mathbf{a} \in \mathbb{R}^m$ the following implication holds: If $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$, then $\mathbf{a}$ satisfies $F$ if and only if there exists a positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models (g \ \varrho \ 0)(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$ for all positive $\eta' \in \mathbb{R}$ smaller than $\eta$. Similarly as with the virtual substitution of $(f, S)$, the assumption that $\mathbf{a}$ satisfies $\gamma$ ensures that $(f, S)\langle\mathbf{a}\rangle$ is defined and yields a real number. The virtual substitution $[x /\!\!/ (f, S) - \varepsilon]$ is defined analogously.

The virtual substitution $[x /\!\!/ (f, S) \pm \varepsilon]$ can be indirectly realized by the virtual substitution $[x /\!\!/ (f, S)]$ as follows: First expand the target atomic formula $g \ \varrho \ 0$ using algorithm `expand-eps-at`, and then apply the virtual substitution $[x /\!\!/ (f, S)]$ to the obtained formula. The intuitive idea behind this expansion algorithm is the following: If, for some parameter values $\mathbf{a} \in \mathbb{R}^m$, the sign of $g\langle\mathbf{a}\rangle$ is nonzero at $\beta \in \mathbb{R}$, then the sign does not change infinitesimally close to $\beta$. However, if $\beta$ is a zero of $g\langle\mathbf{a}\rangle$, then the sign of $g\langle\mathbf{a}\rangle$ infinitesimally close to $\beta$ depends on the signs of the derivatives of $g\langle\mathbf{a}\rangle$ at $\beta$.

Algorithm `expand-eps-at`$(g \ \varrho \ 0, \iota, x)$.
Input: an atomic formula $g \ \varrho \ 0$, where $g = c_d x^d + \cdots + c_1 x + c_0 \in \mathbb{Z}[\mathbf{u}][x]$, $c_i \in \mathbb{Z}[\mathbf{u}]$, and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$. The nonstandard part $\iota$ is $\pm\varepsilon$.
Output: a quantifier-free Tarski formula $\psi(\mathbf{u}, x)$ such that for any parameter values $\mathbf{a} \in \mathbb{R}^m$ and any $\beta \in \mathbb{R}$ we have: $\mathbb{R}^* \models (g \ \varrho \ 0)(\mathbf{a}, \beta + \iota)$ if and only if $\mathbb{R} \models \psi(\mathbf{a}, \beta)$.

1. If $\deg g < 1$, then return $g \ \varrho \ 0$.
2. If $\varrho$ is "=," then return $\bigwedge_{i=0}^{d} c_i = 0$.
3. If $\varrho$ is "$\neq$," then return $\bigvee_{i=0}^{d} c_i \neq 0$.
4. If $\iota = \varepsilon$, then $h := g'$, else $h := -g'$.
5. If $\varrho \in \{<, \leq\}$, then assign "$<$" to $\varsigma$. If $\varrho \in \{\geq, >\}$, then assign "$>$" to $\varsigma$.
6. Return $\big(g \ \varsigma \ 0 \vee (g = 0 \wedge$ `expand-eps-at`$(h \ \varrho \ 0, \iota, x))\big)$.

To prove the correctness of algorithm `expand-eps-at` we will need the following lemma:

**Lemma 30.** *Let $g \in \mathbb{Z}[\mathbf{u}][x]$, and let $\mathbf{a} \in \mathbb{R}^m$ be parameter values such that $g\langle\mathbf{a}\rangle \neq 0$. Let $\beta \in \mathbb{R}$. If $\mathbb{R} \models (g = 0)(\mathbf{a}, \beta)$, then the following hold:*

$$\begin{aligned}
&\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta + \varepsilon) \quad \textit{if and only if} \quad \mathbb{R}^* \models (g' < 0)(\mathbf{a}, \beta + \varepsilon),\\
&\mathbb{R}^* \models (g > 0)(\mathbf{a}, \beta + \varepsilon) \quad \textit{if and only if} \quad \mathbb{R}^* \models (g' > 0)(\mathbf{a}, \beta + \varepsilon),\\
&\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta - \varepsilon) \quad \textit{if and only if} \quad \mathbb{R}^* \models (g' > 0)(\mathbf{a}, \beta - \varepsilon),\\
&\mathbb{R}^* \models (g > 0)(\mathbf{a}, \beta - \varepsilon) \quad \textit{if and only if} \quad \mathbb{R}^* \models (g' < 0)(\mathbf{a}, \beta - \varepsilon).
\end{aligned}$$

*Proof.* We are going to prove that $\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta + \varepsilon)$ is equivalent to the statement $\mathbb{R}^* \models (g' < 0)(\mathbf{a}, \beta + \varepsilon)$. The proof of the three remaining equivalences is similar.

Assume that $\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta + \varepsilon)$. We assume for a contradiction that $\mathbb{R}^* \models (g' \geq 0)(\mathbf{a}, \beta + \varepsilon)$. Our assumption that $g\langle\mathbf{a}\rangle \neq 0$ together with $\mathbb{R} \models (g = 0)(\mathbf{a}, \beta)$ implies that $g\langle\mathbf{a}\rangle$ is a non-constant polynomial with finitely many roots, so $g'\langle\mathbf{a}\rangle$ is a nonzero polynomial with finitely many roots. Lemma 16 ensures that there exists positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models (g' > 0 \wedge g \neq 0)(\mathbf{a}, \beta + \eta')$ for all positive $\eta' \in \mathbb{R}$ strictly smaller than $\eta$. By the mean value theorem we have: For any $\alpha \in \,]\beta, \beta + \eta[$ there exists $\delta \in \,]\beta, \alpha[$ such that $g'\langle\mathbf{a}\rangle(\delta) = \frac{g\langle\mathbf{a}\rangle(\alpha) - g\langle\mathbf{a}\rangle(\beta)}{\alpha - \beta}$. Since $g\langle\mathbf{a}\rangle(\beta) = 0$, $g'\langle\mathbf{a}\rangle(\delta) > 0$, and $\alpha - \beta > 0$, this proves that $\mathbb{R} \models (g > 0)(\mathbf{a}, \alpha)$ for all $\alpha \in \,]\beta, \beta + \eta[$. Using Lemma 16 we obtain $\mathbb{R}^* \models (g > 0)(\mathbf{a}, \beta + \varepsilon)$, which contradicts our assumption $\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta + \varepsilon)$. This shows that $\mathbb{R}^* \models (g' < 0)(\mathbf{a}, \beta + \varepsilon)$ indeed holds.

To prove the converse implication, assume that $\mathbb{R}^* \models (g' < 0)(\mathbf{a}, \beta + \varepsilon)$ and that $\mathbb{R}^* \models (g \geq 0)(\mathbf{a}, \beta + \varepsilon)$. Using similar arguments as above we deduce that there exists positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models (g' < 0 \wedge g \neq 0)(\mathbf{a}, \beta + \eta')$ for all positive $\eta' \in \mathbb{R}$ strictly smaller than $\eta$. By the mean value theorem we again obtain: For any $\alpha \in \,]\beta, \beta + \eta[$ there exists $\delta \in \,]\beta, \alpha[$ such that $g'\langle\mathbf{a}\rangle(\delta) = \frac{g\langle\mathbf{a}\rangle(\alpha) - g\langle\mathbf{a}\rangle(\beta)}{\alpha - \beta}$. Since $g\langle\mathbf{a}\rangle(\beta) = 0$, $g'\langle\mathbf{a}\rangle(\delta) < 0$, and $\alpha - \beta > 0$, this proves that $\mathbb{R} \models (g < 0)(\mathbf{a}, \alpha)$ for any $\alpha \in \,]\beta, \beta + \eta[$. Using Lemma 16 we obtain that $\mathbb{R} \models (g < 0)(\mathbf{a}, \beta + \varepsilon)$. This contradicts our assumption that $\mathbb{R}^* \models (g \geq 0)(\mathbf{a}, \beta + \varepsilon)$. □

**Lemma 31.** *Algorithm* `expand-eps-at` *meets its specification.*

*Proof.* We prove the case when $\iota = \varepsilon$. The proof for $\iota = -\varepsilon$ is similar. We proceed by induction on the degree $d$ of $g$. To begin with, observe that when $d = \deg g < 1$, then `expand-eps-at` trivially meets its specification.

In the sequel we assume that $d > 0$ and that `expand-eps-at` meets its specification for polynomials of degree strictly smaller than $d$. Let $\mathbf{a} \in \mathbb{R}^m$ be arbitrary parameter values and let $\beta \in \mathbb{R}$. We distinguish cases depending on the relation symbol $\varrho$:

- $\varrho$ is "=:" A real univariate polynomial $g\langle\mathbf{a}\rangle$ has a real root $\beta + \varepsilon$ with a nontrivial infinitesimal part if and only if $g\langle\mathbf{a}\rangle$ is identically zero. As a consequence, $g \in \mathbb{Z}[\mathbf{u}][x]$ has a real root with a nontrivial infinitesimal part if and only if $\bigwedge_{i=0}^{d} c_i = 0$ is satisfied by $\mathbf{a}$. Therefore, $\mathbb{R}^* \models (g = 0)(\mathbf{a}, \beta + \varepsilon)$ is equivalent to $\mathbb{R} \models (g = 0)(\mathbf{a})$. This is in turn equivalent to the statement that $\mathbf{a}$ satisfies the formula $\psi$ returned by the algorithm.

- $\varrho$ is "<:" In this case the formula $\psi$ returned by the algorithm is of the form $g < 0 \vee (g = 0 \wedge \psi')$, where $\psi'$ is the formula returned by `expand-eps-at`$(g' < 0, \varepsilon, x)$. Since $\deg g'$ is strictly smaller than $\deg g$, the induction hypothesis ensures that $\mathbb{R}^* \models (g' < 0)(\mathbf{a}, \beta + \varepsilon)$ if and only if $\mathbb{R} \models \psi'(\mathbf{a}, \beta)$. We now have to show the following equivalence:

  $$\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta + \varepsilon) \quad \text{if and only if} \quad \mathbb{R} \models \psi(\mathbf{a}, \beta).$$

  The statement on the right hand side of the equivalence translates to $\mathbb{R} \models (g < 0 \vee (g = 0 \wedge \psi'))(\mathbf{a}, \beta)$, which is obviously equivalent to $\mathbb{R} \models (g \geq 0 \longrightarrow (g = 0 \wedge \psi'))(\mathbf{a}, \beta)$.

  To prove the equivalence, we first assume that $\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta + \varepsilon)$, and show that $\mathbb{R} \models (g \geq 0 \longrightarrow (g = 0 \wedge \psi'))(\mathbf{a}, \beta)$ follows. Let us therefore assume $\mathbb{R} \models (g \geq 0)(\mathbf{a}, \beta)$. We show that $\mathbb{R} \models (g = 0 \wedge \psi')(\mathbf{a}, \beta)$.

  Assume for a contradiction that $\mathbb{R} \models (g > 0)(\mathbf{a}, \beta)$. Then it follows by Lemma 18 (i) that $\mathbb{R}^* \models (g > 0)(\mathbf{a}, \beta + \varepsilon)$. This is a contradiction, because we assume $\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta + \varepsilon)$.

  Therefore we have $\mathbb{R} \models (g = 0)(\mathbf{a}, \beta)$, and Lemma 30 ensures that $\mathbb{R}^* \models (g' < 0)(\mathbf{a}, \beta + \varepsilon)$. This is by the induction hypothesis equivalent to $\mathbb{R} \models \psi'(\mathbf{a}, \beta)$. We have just shown that $\mathbb{R} \models (g = 0 \wedge \psi')(\mathbf{a}, \beta)$, hence $\mathbb{R} \models \big(g < 0 \vee (g = 0 \wedge \psi')\big)(\mathbf{a}, \beta)$.

  To prove the converse implication we assume that $\mathbb{R} \models (g \geq 0 \longrightarrow (g = 0 \wedge \psi'))(\mathbf{a}, \beta)$, i.e., $\mathbb{R} \models (g < 0)(\mathbf{a}, \beta)$ or $\mathbb{R} \models (g = 0 \wedge \psi')(\mathbf{a}, \beta)$. We show that $\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta + \varepsilon)$ follows in both cases:

  - If $\mathbb{R} \models (g < 0)(\mathbf{a}, \beta)$, then $\mathbb{R}^* \models (g < 0)(\mathbf{a}, \beta + \varepsilon)$ follows by Lemma 18 (i).
  - If $\mathbb{R} \models (g = 0 \wedge \psi')(\mathbf{a}, \beta)$, then the induction hypothesis ensures that $\mathbb{R}^* \models (g' < 0)(\mathbf{a}, \beta + \varepsilon)$.

For $\varrho \in \{\neq, \leq, \geq, >\}$ the proof requires a similar case distinction combined with the application of Lemmas 18 and 30 as for the cases "=" and "<." □

We are finally ready to prove that using `expand-eps-at` as a preprocessing step and then carrying out a virtual substitution $[x /\!\!/ (f, S)]$ leads to equivalent results as the direct application of virtual substitution $[x /\!\!/ (f, S) \pm \varepsilon]$.

**Proposition 32.** *Let $g \varrho 0$, where $g \in \mathbb{Z}[\mathbf{u}][x]$ and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$, be an atomic formula. Let $(f, S)$ be a parametric root description of $f \in \mathbb{Z}[\mathbf{u}][x]$. Let $\iota$ be $-\varepsilon$ or $\varepsilon$. Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values satisfying a guard $\gamma$ of $(f, S)$. Then the following holds:*

$$\mathbb{R} \models (g \varrho 0)[x /\!\!/ (f, S) + \iota](\mathbf{a}) \quad \textit{if and only if} \quad \mathbb{R} \models \phi[x /\!\!/ (f, S)](\mathbf{a}),$$

*where $\phi$ is the Tarski formula returned by* `expand-eps-at`$(g \varrho 0, \iota, x)$.

*Proof.* Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values satisfying $\gamma$. We prove the proposition only for the case when $\iota$ is $\varepsilon$, because the proof for $-\varepsilon$ is similar.

Assume that $\mathbf{a}$ satisfies $(g \varrho 0)[x /\!\!/ (f, S) + \iota]$. By the definition of virtual substitution $[x /\!\!/ (f, S) + \iota]$, there exists a positive $\eta \in \mathbb{R}$ such that for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\mathbb{R} \models (g \varrho 0)(\mathbf{a}, (f, S)\langle \mathbf{a} \rangle + \eta')$. By Lemma 16 it follows that $\mathbb{R}^* \models (g \varrho 0)(\mathbf{a}, (f, S)\langle \mathbf{a} \rangle + \varepsilon)$. The specification of algorithm `expand-eps-at` together with Lemma 31 imply that $\mathbb{R} \models \phi(\mathbf{a}, (f, S)\langle \mathbf{a} \rangle)$. Thus, by Theorem 25 (Semantics of Virtual Substitution), it follows that $\mathbb{R} \models \phi[x /\!\!/ (f, S)](\mathbf{a})$.

To prove the converse implication we assume that $\mathbf{a}$ satisfies $\phi[x /\!\!/ (f, S)]$. Using Theorem 25 we obtain that $\mathbb{R} \models \phi(\mathbf{a}, (f, S)\langle \mathbf{a} \rangle)$. The correctness of algorithm `expand-eps-at` ensures that $\mathbb{R}^* \models (g \varrho 0)(\mathbf{a}, (f, S)\langle \mathbf{a} \rangle + \varepsilon)$. By Lemma 16, there exists a positive $\eta \in \mathbb{R}$ such that for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\mathbb{R} \models (g \varrho 0)(\mathbf{a}, (f, S)\langle \mathbf{a} \rangle + \eta')$. Finally, this means that $\mathbf{a}$ satisfies $(g \varrho 0)[x /\!\!/ (f, S) + \varepsilon]$. □

For the sake of an example let us consider polynomial $g = u_3x^3 - u_2x^2 - 5x + u_0 \in \mathbb{Z}[u_0, \ldots, u_3]$. Calling `expand-eps-at`$(g > 0, -\varepsilon, x)$ then yields the following Tarski formula:

$$\begin{aligned} u_3x^3 - u_2x^2 - 5x + u_0 > 0 \vee (u_3x^3 - u_2x^2 - 5x + u_0 = 0 \wedge \\ (-2u_2x + 3u_3x^2 - 5 < 0 \vee (-2u_2x + 3u_3x^2 - 5 = 0 \wedge \\ (-2u_2 + 6u_3x > 0 \vee (-2u_2 + 6u_3x = 0 \wedge 6u_3 < 0))))). \end{aligned}$$

Here note that $g' = -2u_2x + 3u_3x^2 - 5$, $g'' = -2u_2 + 6u_3x$, and $g''' = 6u_3$.

We can see that the virtual substitution of a parametric root description $(f, S)$ plus (or minus) a positive infinitesimal $\varepsilon$ into atomic formula $g \varrho 0$ is expensive: Algorithm `expand-eps-at` expands $g \varrho 0$ in the worst-case to a quantifier-free Tarski formula of depth $d$ containing $2d+1$ atomic formulas taking all the $d$ derivatives of the $d$-th degree polynomial $g$ into account. This means that we have to substitute $(f, S)$ into $2d+1$ atomic formulas. Therefore, it would make sense to somehow shorten the formula computed by the expansion algorithm so that $(f, S)$ would be substituted to a shorter and/or simpler formula. One particularly nice simplification of the expansion—motivated by the specification of algorithm `pseudo-sgn-rem`—would be an algorithm with the following specification:
Input: polynomials $g, f \in \mathbb{Z}[\mathbf{u}][x]$ such that $\deg f > 0$.
Output: a polynomial $h \in \mathbb{Z}[\mathbf{u}][x]$ such that

(i) $\deg h < \deg f$

(ii) For parameter values $\mathbf{a} \in \mathbb{R}^m$ such that $(\operatorname{lc} f)\langle\mathbf{a}\rangle \neq 0$ and any $\beta \in \mathbb{R}$ the following implication holds: If $f\langle\mathbf{a}\rangle(\beta) = 0$, then

$$\operatorname{sgn}\big(g\langle\mathbf{a}\rangle(\beta+\varepsilon)\big) = \operatorname{sgn}\big(h\langle\mathbf{a}\rangle(\beta+\varepsilon)\big).$$

We would then use an algorithm meeting this specification to obtain $h$ with the same sign as $g$ near any root of $f$. Afterwards we would expand $h \varrho 0$. The length of this expanded formula would be independent of the degree of $g$; it would depend exclusively on the degree of $f$. This would simplify the Boolean structure of the expanded formula is some cases. The following example shows that the polynomial $h$ computed by algorithm `pseudo-sgn-rem` does *not* meet our specification:

**Example 33.** Consider polynomials $g = x^2+u_2x+u_3$ and $f = u_0x^2-x+u_1$ from $\mathbb{Z}[u_0,u_1,u_2,u_3][x]$. Then `pseudo-sgn-rem`$(g,f,x)$ yields $h = (u_0+u_0^2u_2)x - u_0u_1+u_0^2u_3$. For parameter values $\mathbf{a} = (-1,2,1,-2)$ we obtain $g\langle\mathbf{a}\rangle = x^2+x-2$, $f\langle\mathbf{a}\rangle = -x^2-x+2$, and $h\langle\mathbf{a}\rangle = 0$. Observe that $(\operatorname{lc} f)\langle\mathbf{a}\rangle \neq 0$ and $f\langle\mathbf{a}\rangle = -g\langle\mathbf{a}\rangle$.

At the same time, $f\langle\mathbf{a}\rangle(\beta_1) = 0$ for $\beta_1 = -2$ but

$$\operatorname{sgn}\big(g\langle\mathbf{a}\rangle(\beta_1+\varepsilon)\big) = -1 \quad \text{and} \quad \operatorname{sgn}\big(h\langle\mathbf{a}\rangle(\beta_1+\varepsilon)\big) = 0.$$

Similarly, $f\langle\mathbf{a}\rangle(\beta_2) = 0$ for $\beta_2 = 1$ but

$$\operatorname{sgn}\big(g\langle\mathbf{a}\rangle(\beta_2+\varepsilon)\big) = 1 \quad \text{and} \quad \operatorname{sgn}\big(h\langle\mathbf{a}\rangle(\beta_2+\varepsilon)\big) = 0.$$

This shows that part (ii) of the specification above does *not* hold for $h$ computed by `pseudo-sgn-rem`. ◇

The problem with polynomial $h$ from Example 33 occurs when the parameters $\mathbf{u}$ are assigned values $\mathbf{a} \in \mathbb{R}^m$ such that $g\langle\mathbf{a}\rangle$ and $f\langle\mathbf{a}\rangle$ have a common real root, and the signs of $g\langle\mathbf{a}\rangle$ and $f\langle\mathbf{a}\rangle$ near this common root differ. If we considered only cases when $f\langle\mathbf{a}\rangle$ and $g\langle\mathbf{a}\rangle$ cannot have a common real root, then we would be on a safe side: In fact, if the resultant of $g$ and $f$ is nonzero for any parameter values $\mathbf{a} \in \mathbb{R}^m$, then $h$ computed by `pseudo-sgn-rem`$(g,f,x)$ meets both parts of our specification above. If the resultant of $g$ and $f$ has a real root, then the following example shows that not only algorithm `pseudo-sgn-rem` does not compute $h$ meeting the desired specification, but no polynomial meeting the specification exists.

**Example 34.** Consider polynomials $g = u_0x+1$ and $f = x-1$ from $\mathbb{Z}[u_0][x]$. We search for a polynomial $h \in \mathbb{Z}[u_0][x]$ with the following properties:

(i) $\deg h < \deg f = 1$, i.e., $h \in \mathbb{Z}[u_0]$

(ii) For any parameter value $a \in \mathbb{R}$ such that $(\operatorname{lc} f)\langle\mathbf{a}\rangle \neq 0$ and any $\beta \in \mathbb{R}$ the following implication holds: If $f\langle a\rangle(\beta) = 0$, then

$$\operatorname{sgn}\big(g\langle a\rangle(\beta+\varepsilon)\big) = \operatorname{sgn}\big(h\langle a\rangle(\beta+\varepsilon)\big).$$

Assume for a contradiction that there exists $h \in \mathbb{Z}[u_0]$ meeting (ii).

To begin with, observe that for any parameter value $a \in \mathbb{R}$ the following hold: $(\operatorname{lc} f)\langle \mathbf{a} \rangle \neq 0$, and $f\langle a \rangle(\beta) = 0$ implies that $\beta = 1$, because $f = x - 1$. Moreover, $(f, S)$, where $S = \big\{((-1, 0, 1), 1)\big\}$, is a parametric root description with guard "true" such that $(f, S)\langle a \rangle = \beta = 1$ for any parameter value $a \in \mathbb{R}$. Without loss of generality we therefore assume that $\beta = 1$.

Observe that for any parameter value $a \in \mathbb{R}$ we have $\operatorname{sgn}\big(g\langle a \rangle(\beta + \varepsilon)\big) = 0$ if and only if $\mathbb{R}^* \models (g = 0)(a, \beta + \varepsilon)$. Since $(f, S)\langle a \rangle = \beta$ for any parameter value $a \in \mathbb{R}$, this is equivalent to $\mathbb{R}^* \models (g = 0)(a, (f, S)\langle a \rangle + \varepsilon)$. Using Algorithm `expand-eps-at`$(g = 0, \varepsilon, x)$ we obtain that this is the case if and only if

$$\mathbb{R} \models \big(u_0 = 0 \wedge 1 = 0\big)[x /\!\!/ (f, S)](a).$$

This is obviously equivalent to $\mathbb{R} \models \big(\text{false}\big)[x /\!\!/ (f, S)](a)$. This means that $\operatorname{sgn}\big(g\langle a \rangle(\beta + \varepsilon)\big) = 0$ does not hold for any $a \in \mathbb{R}$. By (ii) we then obtain that $\operatorname{sgn}\big(h\langle a \rangle(\beta + \varepsilon)\big) \neq 0$ for all $a \in \mathbb{R}$. Since $h$ does not contain variable $x$, this ensures that $h\langle a \rangle \neq 0$ for any $a \in \mathbb{R}$.

At the same time, observe that $\operatorname{sgn}\big(g\langle a \rangle(\beta + \varepsilon)\big) = 1$ holds if and only if $\mathbb{R}^* \models (g > 0)(a, \beta + \varepsilon)$. This is equivalent to $\mathbb{R}^* \models (g > 0)(a, (f, S)\langle a \rangle + \varepsilon)$, because $(f, S)\langle a \rangle = \beta$. Using Algorithm `expand-eps-at`$(g > 0, \varepsilon, x)$ we obtain that this is equivalent to

$$\mathbb{R} \models \big(u_0 x + 1 > 0 \vee (u_0 x + 1 = 0 \wedge u_0 > 0)\big)[x /\!\!/ (f, S)](a).$$

Carrying out the virtual substitution of a rational as we described it in the beginning of this section we obtain

$$\mathbb{R} \models \big(u_0 + 1 > 0 \vee (u_0 + 1 = 0 \wedge u_0 > 0)\big)(a).$$

Simplification of the formula yields an equivalent statement $\mathbb{R} \models (u_0 + 1 > 0)(a)$. Picking $a = 7$ obviously satisfies $u_0 + 1 > 0$, and we deduce that $h\langle 7 \rangle$ is positive, because $\operatorname{sgn}\big(g\langle a \rangle(\beta + \varepsilon)\big) = \operatorname{sgn}\big(h\langle a \rangle(\beta + \varepsilon)\big)$ for any $a \in \mathbb{R}$.

Similarly, we have $\operatorname{sgn}\big(g\langle a \rangle(\beta + \varepsilon)\big) = -1$ if and only if $\mathbb{R}^* \models (g < 0)(a, \beta + \varepsilon)$. Using the fact that $(f, S)\langle a \rangle = \beta$ for any $a \in \mathbb{R}$, we obtain that this holds if and only if $\mathbb{R}^* \models (g < 0)(a, (f, S)\langle a \rangle + \varepsilon)$. Algorithm `expand-eps-at`$(g < 0, \varepsilon, x)$ then yields an equivalent condition

$$\mathbb{R} \models \big(u_0 x + 1 < 0 \vee (u_0 x + 1 = 0 \wedge u_0 < 0)\big)[x /\!\!/ (f, S)](a),$$

which simplifies to $\mathbb{R} \models (u_0 + 1 \leq 0)(a)$ after carrying out the virtual substitution of a rational. Parameter value $a = -5$ obviously satisfies $u_0 + 1 \leq 0$, so using (ii) we deduce that $h\langle -5 \rangle$ is negative.

We have just shown that $h\langle 7 \rangle$ is positive and $h\langle -5 \rangle$ is negative. Since $h$ is continuous, there exists a real root of $h$ in the interval $]-5, 7[$; a contradiction.

An informal description of the problem here is rather simple: The polynomial $u_0(1 + \varepsilon) + 1 = u_0 + \varepsilon u_0 + 1$—obtained by substituting $1 + \varepsilon$ into $g$—does not have a real root, but, at the same time, changes its sign in the real interval $]-5, 7[$. This cannot be modeled by a polynomial from $\mathbb{Z}[u_0]$. Our formalism makes this intuition precise and shows that this is indeed the case. $\diamond$

The mappings $[x /\!\!/ \pm\infty]$ and $[x /\!\!/ (f, S) \pm \varepsilon]$ introduced in this section naturally generalize to any quantifier-free Tarski formula $\varphi$, so by $\varphi[x /\!\!/ \pm\infty]$ and

$\varphi[x /\!\!/ (f,S) \pm \varepsilon]$ we denote the formula obtained by applying the respective virtual substitution to each atomic formula occurring in $\varphi$. In analogy to Theorem 25 we show that this generalization is indeed compatible with our definitions of virtual substitutions involving nonstandard symbols:

**Theorem 35** (Semantics of Virtual Substitution with Nonstandard Symbols)**.** *Let $\varphi(\mathbf{u}, x)$ be a quantifier-free Tarski formula. Then the following hold:*

*(i) Let $(f,S)$ be a parametric root description. Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values satisfying a guard $\gamma$ of $(f,S)$. Then $\mathbb{R} \models \varphi[x /\!\!/ (f,S) \pm \varepsilon](\mathbf{a})$ if and only if there exists positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle \pm \eta')$ holds for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$.*

*(ii) $\mathbb{R} \models \varphi[x /\!\!/ \pm\infty](\mathbf{a})$ if and only if there exists $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \eta')$ holds for any $\eta' \in \mathbb{R}$ greater/smaller than $\eta$.*

*Proof.* (i) We prove this part of the theorem only for $(f,S) + \varepsilon$. The proof for $(f,S) - \varepsilon$ is similar, so it is omitted.

Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values satisfying $\gamma$, i.e., $(f,S)\langle\mathbf{a}\rangle$ yields a real number. We proceed by structural induction on the Boolean structure of $\varphi$. To begin with, observe that if $\varphi$ is an atomic formula, then the theorem directly follows from our definition of virtual substitution $[x /\!\!/ (f,S) + \varepsilon]$.

Assume now that $\varphi$ is of the form $\varphi_1 \wedge \varphi_2$, and that the theorem holds for $\varphi_1$ and $\varphi_2$. Since we defined that $\varphi[x /\!\!/ (f,S) + \varepsilon]$ is obtained by applying $[x /\!\!/ (f,S) + \varepsilon]$ to each atomic formula of $\varphi$, we have $\varphi[x /\!\!/ (f,S) + \varepsilon] = \varphi_1[x /\!\!/ (f,S) + \varepsilon] \wedge \varphi_2[x /\!\!/ (f,S) + \varepsilon]$. By the induction hypothesis, for $i \in \{1,2\}$ we have: $\mathbb{R} \models \varphi_i[x /\!\!/ (f,S) + \varepsilon](\mathbf{a})$ if and only if there exists positive $\eta_i \in \mathbb{R}$ such that for any positive $\eta_i' \in \mathbb{R}$ smaller than $\eta_i$ we have $\mathbb{R} \models \varphi_i(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle + \eta_i')$. Now we need to show that $\mathbb{R} \models \varphi[x /\!\!/ (f,S) + \varepsilon](\mathbf{a})$ if and only if there exists positive $\eta \in \mathbb{R}$ such that for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\mathbb{R} \models \varphi(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle + \eta')$.

First assume that $\mathbb{R} \models (\varphi_1 \wedge \varphi_2)[x /\!\!/ (f,S) + \varepsilon](\mathbf{a})$ holds. By the induction hypothesis, for $\eta = \min\{\eta_1, \eta_2\}$ we have: $\mathbb{R} \models \varphi_i(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle + \eta')$ holds for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$, i.e., $\mathbb{R} \models (\varphi_1 \wedge \varphi_2)(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle + \eta')$ holds for any positive $\eta' \in \mathbb{R}$. This proves one implication.

To prove the converse implication, we assume that there exists $\eta \in \mathbb{R}$ such that for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\mathbb{R} \models (\varphi_1 \wedge \varphi_2)(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle + \eta')$. The induction hypothesis then yields $\mathbb{R} \models (\varphi_1 \wedge \varphi_2)[x /\!\!/ (f,S) + \varepsilon](\mathbf{a})$. This finishes the proof for the case when $\varphi$ is of the form $\varphi_1 \wedge \varphi_2$.

We continue with the case when $\varphi$ is of the form $\neg\varphi'$. Assume that the theorem holds for $\varphi'$. Similarly as in the previous case, we have $\varphi[x /\!\!/ (f,S) + \varepsilon] = \neg(\varphi'[x /\!\!/ (f,S) + \varepsilon])$, and the induction hypothesis ensures that $\mathbb{R} \models \varphi'[x /\!\!/ (f,S) + \varepsilon](\mathbf{a})$ holds if and only if there exists positive $\eta \in \mathbb{R}$ such that for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\mathbb{R} \models \varphi'(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle + \eta')$. We need to prove that $\mathbb{R} \models \varphi[x /\!\!/ (f,S) + \varepsilon](\mathbf{a})$ if and only if there exists positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, (f,S)\langle\mathbf{a}\rangle + \eta')$ holds for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$.

First assume that $\mathbb{R} \models \varphi[x /\!\!/ (f,S) + \varepsilon](\mathbf{a})$ holds. Therefore, we obtain $\mathbb{R} \models (\neg\varphi')[x /\!\!/ (f,S) + \varepsilon](\mathbf{a})$, i.e., $\mathbb{R} \not\models \varphi'[x /\!\!/ (f,S) + \varepsilon](\mathbf{a})$ holds. By

the induction hypothesis we obtain: For all positive $\eta \in \mathbb{R}$ there exists a positive $\eta' \in \mathbb{R}$ smaller than $\eta$ such that $\mathbb{R} \models \neg\varphi'(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$, i.e., $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$ holds. By Lemma 16 this is equivalent to the statement "There exists a positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$." This finishes the proof of one implication.

Now assume that there exists positive $\eta \in \mathbb{R}$ such that for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$, i.e., $\mathbb{R} \not\models \varphi'(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$. By the induction hypothesis we obtain: For all positive $\eta \in \mathbb{R}$ there exists $\eta' \in \mathbb{R}$ smaller than $\eta$ such that we have $\mathbb{R} \models \neg\varphi'(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$, i.e., $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$. Again, Lemma 16 ensures that this is equivalent to "There exists a positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$," and the proof of the converse implication is finished.

The proof is similar for other Boolean operators, so we omit it.

(ii) The proof is done along the same lines as the proof of (i)—but instead of using Lemma 16 we use Lemma 17—so we omit it. □

## 2.4 Quantifier Elimination Algorithm Scheme

In this section we give a scheme for quantifier elimination algorithms based on virtual substitution. The scheme is presented as algorithm `vs-scheme`, which is parameterized by three precisely specified sub-algorithms. An instantiation of the scheme is obtained by providing concrete sub-algorithms meeting the specifications and "plugging" them into their places in the scheme. Each instantiation yields a quantifier elimination algorithm using virtual substitution. The scheme is indeed based on the notions of parametric root descriptions, candidate solutions, and virtual substitution introduced in previous sections. Our main aim here is to prove the correctness of the scheme as a whole, i.e., *any* algorithm obtained by instantiation of the scheme is correct and its complexity is given as a function of complexities of the provided sub-algorithms.

We begin our exposition by specifying the three mentioned sub-algorithms:

1. Algorithm `at-cs`$(f \ \varrho \ 0, x)$:
   Input: an atomic formula $f \ \varrho \ 0$, where $f \in \mathbb{Z}[\mathbf{u}][x]$ and the relation $\varrho$ is one of $\{=, \neq, <, \leq, \geq, >\}$.
   Output: a set of candidate solutions for $f \ \varrho \ 0$.

2. Algorithm `guard`$\big((f, S), x\big)$:
   Input: a parametric root description $(f, S)$, where $f \in \mathbb{Z}[\mathbf{u}][x]$ and $S$ is a finite set $\{(t_1, r_1), \ldots, (t_k, r_k)\}$ of root specifications of $f$.
   Output: a quantifier-free guard of $(f, S)$.

3. Algorithm `vs-prd-at`$\big(g \ \varrho \ 0, (f, S), x\big)$:
   Input: an atomic formula $g \ \varrho \ 0$ and a parametric root description $(f, S)$ such that $g$, $f \in \mathbb{Z}[\mathbf{u}][x]$, $\varrho \in \{=, \neq, <, \leq, \geq, >\}$, and $\deg g < \deg f$.
   Output: $(g \ \varrho \ 0)[x \mathbin{/\!\!/} (f, S)]$, i.e., a quantifier-free formula in the parameters $\mathbf{u}$ meeting our specification of virtual substitution.

Note that the specification of algorithm `vs-prd-at` was also given in Section 2.3. We repeat the definition here for completeness; all results of Section 2.3 remain valid. In the rest of this section we assume that the mentioned algorithms exist and regard them as black-boxes.

We continue with a description of algorithm `vs-at`. This algorithm will be called by our scheme to carry out a virtual substitution of a single *test point* into a single atomic formula.

Algorithm `vs-at`$(g\ \varrho\ 0, e, x)$.
Input: an atomic formula $g\ \varrho\ 0$, where $g \in \mathbb{Z}[\mathbf{u}][x]$ and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$, test point $e$ is one of the following:

(a) parametric root description $(f, S)$, where $f \in \mathbb{Z}[\mathbf{u}][x]$,

(b) parametric root description plus/minus a positive infinitesimal $(f, S) \pm \varepsilon$, where $f \in \mathbb{Z}[\mathbf{u}][x]$,

(c) nonstandard symbol $\pm\infty$.

Output: a quantifier-free Tarski formula $\psi$ equivalent to $(g\ \varrho\ 0)[x \mathbin{/\!\!/} e]$.

1. If $e$ is $(f, S)$, then

   1.1. $h :=$ `pseudo-sgn-rem`$(g, f, x)$

   1.2. Return `vs-prd-at`$\big(h\ \varrho\ 0, (f, S), x\big)$.

2. If $e$ is $(f, S) + \iota$, where $\iota$ is $\pm\varepsilon$, then

   2.1. $\phi :=$ `expand-eps-at`$(g\ \varrho\ 0, \iota, x)$

   2.2. Replace each atomic formula $h\ \varrho\ 0$ occurring in $\phi$ with quantifier-free formula `vs-at`$(h\ \varrho\ 0, (f, S), x)$, and return the resulting formula obtained this way.

3. If $e$ is $\pm\infty$, then

   3.1. Return `vs-inf-at`$(g\ \varrho\ 0, e, x)$.

The correctness of algorithm `vs-at` is a straightforward consequence of the results of Section 2.3:

**Lemma 36.** *Algorithm* `vs-at` *meets its specification.*

*Proof.* We distinguish three cases depending on the type of test point $e$:

1. If $e$ is $(f, S)$, then the correctness of `vs-at` follows from Proposition 27.

2. If $e$ is $(f, S) + \iota$, where $\iota$ is $\pm\varepsilon$, then the correctness of `vs-at` follows from Proposition 32.

3. If $e$ is $\pm\infty$, then the correctness of `vs-at` follows from Proposition 29. □

Now we are ready to present our algorithm scheme. The scheme proceeds as follows: Using algorithm `at-cs` it first computes a set of candidate solutions $\mathfrak{c}$ for an input Tarski formula $\varphi$. Afterwards, this set of candidate solutions is converted into a set of test points by selecting appropriate parametric root descriptions from $\mathfrak{c}$, adjusting them by adding a positive infinitesimal, if needed.

Finally, each test point obtained this way is substituted into the input formula $\varphi$ by applying algorithm `vs-at` to each atomic formula of $\varphi$.

Algorithm `vs-scheme`$(\varphi, x)$.
Input: a quantifier-free Tarski formula $\varphi(\mathbf{u}, x)$, which is an $\wedge$-$\vee$-combination of atomic formulas, a variable $x$.
Output: a quantifier-free Tarski formula $\psi(\mathbf{u})$ equivalent to $\exists x(\varphi)$.

1. Extract from $\varphi$ the set $A$ of all atomic formulas containing $x$.
2. $\mathfrak{c} := \emptyset$
3. For each atomic formula $f \varrho 0$ in $A$ do
   3.1. $\mathfrak{c} := \mathfrak{c} \cup$ `at-cs`$(f \varrho 0, x)$
4. $E := \{-\infty\}$
5. For each candidate solution $(f, S, \tau) \in \mathfrak{c}$ do
   5.1. If $\tau$ is "IP" or "WLB," then add $(f, S)$ to $E$.
   5.2. If $\tau$ is "EP" or "SLB," then add $(f, S) + \varepsilon$ to $E$.
6. $\psi :=$ false
7. For each test point $e \in E$ do
   7.1. If $e$ is $(f, S)$, then
      7.1.1. Copy $\varphi$ to $\varphi'$.
      7.1.2. $\gamma :=$ `guard`$\big((f, S), x\big)$
      7.1.3. Compute $\varphi'[x /\!\!/ (f, S)]$ by replacing each atom $g \varrho 0$ occurring in $\varphi'$ with quantifier-free formula `vs-at`$(g \varrho 0, (f, S), x)$.
      7.1.4. $\psi := \psi \vee \big(\gamma \wedge \varphi'[x /\!\!/ (f, S)]\big)$
   7.2. If $e$ is $(f, S) + \varepsilon$, then
      7.2.1. Copy $\varphi$ to $\varphi'$.
      7.2.2. $\gamma :=$ `guard`$\big((f, S), x\big)$
      7.2.3. Compute $\varphi'[x /\!\!/ (f, S) + \varepsilon]$ by replacing each atom $g \varrho 0$ occurring in $\varphi'$ with quantifier-free formula `vs-at`$\big(g \varrho 0, (f, S) + \varepsilon, x\big)$.
      7.2.4. $\psi := \psi \vee \big(\gamma \wedge \varphi'[x /\!\!/ (f, S) + \varepsilon]\big)$
   7.3. If $e$ is $-\infty$, then
      7.3.1. Copy $\varphi$ to $\varphi'$.
      7.3.2. Compute $\varphi'[x /\!\!/ -\infty]$ by replacing each atom $g \varrho 0$ occurring in $\varphi'$ with quantifier-free formula `vs-at`$(g \varrho 0, -\infty, x)$.
      7.3.3. $\psi := \psi \vee \varphi'[x /\!\!/ -\infty]$
8. Return $\psi$.

**Theorem 37.** *Algorithm* `vs-scheme` *meets its specification.*

*Proof.* We have to prove that for any parameter values $\mathbf{a} \in \mathbb{R}^m$ the following equivalence holds: $\mathbb{R} \models \psi(\mathbf{a})$ if and only if there exists $\beta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \beta)$, i.e., the satisfying set $\Phi(\varphi, \mathbf{a})$ is nonempty. Let therefore $\mathbf{a} \in \mathbb{R}^m$ be arbitrary parameter values. We show that $\mathbf{a}$ satisfies $\psi$ if and only if the satisfying set $\Phi(\varphi, \mathbf{a})$ is nonempty.

Assume first that $\mathbf{a}$ satisfies $\psi$. Observe that $\psi$ returned by `vs-scheme` is a quantifier-free disjunction obtained in step 7 by consecutively substituting all the test points from $E$ into $\varphi$. There are three cases to consider:

1. $\mathbf{a}$ satisfies $\gamma \wedge \varphi[x /\!\!/ (f, S)]$ for some test point $(f, S) \in E$: Since `vs-at` meets its specification, and $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$, we use Theorem 25 to deduce that $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle)$, i.e., $\Phi(\varphi, \mathbf{a})$ is nonempty.

2. $\mathbf{a}$ satisfies $\gamma \wedge \varphi[x /\!\!/ (f, S) + \varepsilon]$ for some test point $(f, S)+\varepsilon \in E$: Similarly to the previous case, since `vs-at` meets its specification, and $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$, we use Theorem 35 (i) to deduce that there exists a positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$, i.e., $\Phi(\varphi, \mathbf{a})$ is obviously nonempty.

3. $\mathbf{a}$ satisfies $\varphi[x /\!\!/ -\infty]$: Since `vs-at` meets its specification, we use Theorem 35 (ii) to deduce that there exists $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \eta')$ for any $\eta' \in \mathbb{R}$ smaller than $\eta$, i.e., $\Phi(\varphi, \mathbf{a})$ is unbounded from below and in particular nonempty.

Now we prove the converse implication. Assume that the satisfying set $\Phi(\varphi, \mathbf{a})$ is nonempty. Since `at-cs` returns a set of candidate solutions for an atomic formula, and $\varphi$ is an $\wedge$-$\vee$-combination of atomic formulas, Proposition 15 ensures that the set $\mathfrak{c}$ computed by algorithm `vs-scheme` in step 3 is a set of candidate solutions for $\varphi$. We assume that $\Phi(\varphi, \mathbf{a}) \neq \emptyset$, so Theorem 19 (i) ensures that there exists $\xi \in \mathcal{L}$ such that $\mathbb{R}^* \models \varphi(\mathbf{a}, \xi)$. According to the definition of the set $\mathcal{L}$, now there are three cases to consider:

1. There exists $\xi \in \mathcal{L}$ such that $\xi = (f, S, \tau)\langle\mathbf{a}\rangle$ for some $(f, S, \tau) \in \mathfrak{c}$, where $\tau$ is "IP" or "WLB." In this case we have $\mathbb{R}^* \models \varphi(\mathbf{a}, \xi)$ and in particular $\mathbb{R} \models \varphi(\mathbf{a}, \xi)$, because $\xi \in \mathbb{R}$ and $\mathbb{R}^*$ is an extension field of $\mathbb{R}$. Since $\xi = (f, S, \tau)\langle\mathbf{a}\rangle$, $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$. Theorem 25 then guarantees that $\mathbb{R} \models \varphi[x /\!\!/ (f, S)](\mathbf{a})$, so we obtain that $\mathbb{R} \models (\gamma \wedge \varphi[x /\!\!/ (f, S)])(\mathbf{a})$. Now observe that $(f, S) \in E$, because $\tau$ is "IP" or "WLB." Thus, $(f, S)$ was substituted into $\varphi$ in step 7.1. This implies that $\mathbb{R} \models \psi(\mathbf{a})$, because $\psi$ is a disjunction containing $\gamma \wedge \varphi[x /\!\!/ (f, S)]$ as a disjunct.

2. There exists $\xi \in \mathcal{L}$ such that $\xi = (f, S, \tau)\langle\mathbf{a}\rangle + \varepsilon$ for some $(f, S, \tau) \in \mathfrak{c}$, where $\tau$ is "EP" or "SLB." In this case we have $\mathbb{R}^* \models \varphi(\mathbf{a}, \xi+\varepsilon)$. Lemma 16 ensures that there exists a positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \xi+\eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$. Since $\xi = (f, S, \tau)\langle\mathbf{a}\rangle$, $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$. Theorem 35 (i) then implies that $\mathbb{R} \models \varphi[x /\!\!/ (f, S) + \varepsilon](\mathbf{a})$, so we obtain that $\mathbb{R} \models (\gamma \wedge \varphi[x /\!\!/ (f, S) + \varepsilon])(\mathbf{a})$. Now observe that $(f, S)+\varepsilon \in E$, because $\tau$ is "EP" or "SLB." Thus, $(f, S)+\varepsilon$ was substituted into the input formula $\varphi$ in step 7.2. This implies that $\mathbb{R} \models \psi(\mathbf{a})$, because $\psi$ contains $\gamma \wedge \varphi[x /\!\!/ (f, S) + \varepsilon]$ as a disjunct.

3. We have $\xi \in \mathcal{L}$ for $\xi = -\infty$. In this case we have $\mathbb{R}^* \models \varphi(\mathbf{a}, -\infty)$. Lemma 17 ensures that there exists $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \eta')$ for

any $\eta' \in \mathbb{R}$ strictly smaller than $\eta$. Theorem 35 (ii) then implies that $\mathbb{R} \models \varphi[x /\!\!/ -\infty]$. Since $-\infty \in E$, we know that $-\infty$ was substituted into $\varphi$ in step 7.3 of the algorithm. Again, this implies that $\mathbb{R} \models \psi(\mathbf{a})$, because the formula $\psi$ contains $\gamma \wedge \varphi[x /\!\!/ -\infty]$ as a disjunct.

We have shown that in each of the three cases $\mathbf{a}$ satisfies the output formula $\psi$, so the proof of the theorem is finished. □

The crucial idea behind the proof of Theorem 37 is the following: Since $\mathfrak{c}$ is a set of candidate solutions for $\varphi$, the computed test points together with $-\infty$ are guaranteed to intersect a nonempty satisfying set $\Phi(\varphi, \mathbf{a})$ for any parameter values $\mathbf{a} \in \mathbb{R}^m$. To this end observe that the set $E$ of test points obtained in step 5 is generated only from those candidate solutions that are necessary to use Theorem 19 (i). If we wanted to use part (ii) of Theorem 19, we would have to remove candidate solutions with tag "EP" and add candidate solutions with tag "WLB" in step 5.2 of the algorithm. This would make sense when there were many excluded points, but only a few weak lower bounds candidate solutions.

Observe that the set $E$ of test points obtained in step 5 is an *elimination set for $\varphi$ and $x$* in the following sense: Substituting all test points from $E$ into $\varphi$ using virtual substitution yields a quantifier-free equivalent of $\exists x(\varphi)$. Algorithm `vs-scheme` adds to set $E$ all parametric root descriptions that possibly represent a lower bound of a satisfying set. It is straightforward to adjust the algorithm and its proof to use those parametric root descriptions that possibly represent an upper bound of a satisfying set instead—leading to a different elimination set for $\varphi$ and $x$. Moreover, it is also correct to first analyze the set of candidate solutions $\mathfrak{c}$, and then decide whether the upper or the lower bounds parametric root descriptions should be used. This corresponds to the idea of *bound selection* introduced in [85].

A careful inspection of algorithm `vs-scheme` reveals that the algorithms `guard` and `vs-prd-at` need to handle only such parametric root descriptions $(f, S)$ that are possibly produced as candidate solutions by `vs-at`. This in particular means that for a fixed degree $d$, `vs-prd-at` need not be specified for all possible sets $S$ of root specifications of a polynomial $f$ with $\deg f \leq d$, but only for such root specification sets that represent a candidate solution returned by `vs-at`. We will extensively take advantage of this observation when instantiating the scheme in Section 2.5.

Later in Section 2.6 we will show that all existing quantifier elimination algorithms based on virtual substitution can be obtained by an appropriate instantiation of our scheme.

## 2.5 Instantiating the Scheme

Now we change our abstract perspective and discuss how to instantiate the algorithm scheme of Section 2.4. Our aim here is to obtain self-contained quantifier elimination algorithms for degree at most three. We will also prove that our approach is extensible beyond this degree bound.

### 2.5.1 Linear Virtual Substitution

Let us illustrate the instantiation process on the case when the variable $x$ to be eliminated occurs only linearly in the input formula $\varphi(\mathbf{u}, x)$.

There exist only two real 1-types, namely $(-1, 0, 1)$ and $(1, 0, -1)$, so a polynomial $f = ax + b$, where $a$, $b \in \mathbb{Z}[\mathbf{u}]$ has at most two realizable real 1-types. It is also not hard to see that $f\langle\mathbf{a}\rangle$ is of real type $(-1, 0, 1)$ if and only if $\mathbb{R} \models (a > 0)(\mathbf{a})$, and $f\langle\mathbf{a}\rangle$ is of real type $(1, 0, -1)$ if and only if $\mathbb{R} \models (a < 0)(\mathbf{a})$. With these observations in mind we can directly write correct realizations of the three sub-algorithms for the linear case:

Algorithm `at-cs-1`$(f \ \varrho \ 0, x)$:
Input: an atomic formula $f \ \varrho \ 0$, where $f \in \mathbb{Z}[\mathbf{u}][x]$ has degree one and the relation $\varrho$ is one of $\{=, \neq, <, \leq, \geq, >\}$.
Output: a set of candidate solutions for $f \ \varrho \ 0$.

1. If $\varrho$ is "$=$," then $\mathfrak{c} := \big\{\big(f, ((-1, 0, 1), 1), \mathrm{IP}\big), \big(f, ((1, 0, -1), 1), \mathrm{IP}\big)\big\}$.
2. If $\varrho$ is "$\neq$," then $\mathfrak{c} := \big\{\big(f, ((-1, 0, 1), 1), \mathrm{EP}\big), \big(f, ((1, 0, -1), 1), \mathrm{EP}\big)\big\}$.
3. If $\varrho$ is "$<$," then $\mathfrak{c} := \big\{\big(f, ((-1, 0, 1), 1), \mathrm{SUB}\big), \big(f, ((1, 0, -1), 1), \mathrm{SLB}\big)\big\}$.
4. If $\varrho$ is "$\leq$," then $\mathfrak{c} := \big\{\big(f, ((-1, 0, 1), 1), \mathrm{WUB}\big), \big(f, ((1, 0, -1), 1), \mathrm{WLB}\big)\big\}$.
5. If $\varrho$ is "$\geq$," then $\mathfrak{c} := \big\{\big(f, ((-1, 0, 1), 1), \mathrm{WLB}\big), \big(f, ((1, 0, -1), 1), \mathrm{WUB}\big)\big\}$.
6. If $\varrho$ is "$>$," then $\mathfrak{c} := \big\{\big(f, ((-1, 0, 1), 1), \mathrm{SLB}\big), \big(f, ((1, 0, -1), 1), \mathrm{SUB}\big)\big\}$.
7. Return $\mathfrak{c}$.

Algorithm `guard-1`$\big((f, S), x\big)$:
Input: a parametric root description $(f, S)$, where $f \in \mathbb{Z}[\mathbf{u}][x]$ has degree one.
Output: a quantifier-free guard of $(f, S)$.

1. If $S = \big\{\big((-1, 0, 1), 1\big)\big\}$, then return $\operatorname{lc} f > 0$.
2. If $S = \big\{\big((1, 0, -1), 1\big)\big\}$, then return $\operatorname{lc} f < 0$.

Algorithm `vs-prd-at-1`$\big(g \ \varrho \ 0, (f, S), x\big)$:
Input: an atomic formula $g \ \varrho \ 0$ and a parametric root description $(f, S)$ such that $g$, $f \in \mathbb{Z}[\mathbf{u}][x]$ and $\deg g < \deg f = 1$, i.e., $g \in \mathbb{Z}[\mathbf{u}]$.
Output: $(g \ \varrho \ 0)[x \mathbin{/\!\!/} (f, S)]$, i.e., a quantifier-free Tarski formula.

1. Return $g \ \varrho \ 0$.

All we need now is to prove that these three algorithms meet their specifications.

**Theorem 38.** *Algorithms* `at-cs-1`, `guard-1`, *and* `vs-prd-at-1` *meet their specifications.*

*Proof.* Consider a polynomial $f = ax + b$, where $f = ax + b$ and $a$, $b \in \mathbb{Z}[\mathbf{u}]$. Recall that $f$ has at most two realizable real 1-types: $(-1, 0, 1)$ and $(1, 0, -1)$. Guards of these types are $a > 0$ and $a < 0$, respectively, so the correctness of `guard-1` follows.

To prove the correctness of `at-cs-1`, we need to prove that the returned set $\mathfrak{c}$ is a set of candidate solutions for the input formula $f \varrho 0$. We prove this only for $f < 0$, because the proof for the other relation symbols is similar. Observe that for any parameter values $\mathbf{a} \in \mathbb{R}^m$ we have $a\langle\mathbf{a}\rangle > 0$, $a\langle\mathbf{a}\rangle < 0$, or $a\langle\mathbf{a}\rangle = 0$. In the first case the first real root of $f\langle\mathbf{a}\rangle$—with real 1-type $(-1, 0, 1)$—is the only boundary point "strict upper bound" of the satisfying set $\Phi(f < 0, \mathbf{a})$. In the second case the first root of $f\langle\mathbf{a}\rangle$ is the only boundary point "strict lower bound" of $\Phi(f < 0, \mathbf{a})$. In the third case $a\langle\mathbf{a}\rangle = 0$, so there exists no boundary point of the set $\Phi(f < 0, \mathbf{a})$, which is either $\emptyset$ or $\mathbb{R}$. This proves that the set $\{(f, ((-1, 0, 1), 1), \mathrm{SUB}), (f, ((1, 0, -1), 1), \mathrm{SLB})\}$ returned by `at-cs-1`$(f < 0, x)$ is indeed a set of candidate solutions for $f < 0$.

Finally, the correctness of `vs-prd-at-1` follows from the fact that $g$ does not contain $x$, so $g \varrho 0$ is the correct result of substituting $(f, S)$ for $x$ into $g \varrho 0$ by means of virtual substitution as we defined it in Section 2.3. □

Theorem 38 together with the correctness of algorithm `vs-scheme` ensure that plugging these three algorithms into `vs-scheme` yields a correct quantifier elimination algorithm for formulas $\exists x(\varphi(\mathbf{u}, x))$ where $x$ occurs linearly in $\varphi$.

### 2.5.2 Quadratic and Cubic Virtual Substitution

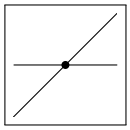
Here we assume that the degree of the variable $x$ to be eliminated from $\varphi(\mathbf{u}, x)$ is at most three. Before we discuss the realizations of algorithms `at-cs`, `guard`, and `vs-prd-at` let us first investigate all realizable real $d$-types for $d \in \{1, 2, 3\}$.

Let $f \in \mathbb{Z}[\mathbf{u}][x]$ be a polynomial of degree $d \in \{1, 2, 3\}$. Recall that each realizable real $d$-type of $f$ is indeed a real $d$-type, so Table 2.1 ensures that there exist at most 2, 6, and 8 realizable real $d$-types of $f$ for $d \in \{1, 2, 3\}$. We list here all relevant real $d$-types along with their respective guards for $f$. Lemma 6 implies that for each "positive" real $d$-type there exists a "negative" real $d$-type. Below we list the real types according to this "positive-negative" pattern; first all real types for the case when $\operatorname{lc} f$ is positive are listed, and then we list the "negative" counterpart for each of them, i.e., the real type of $-f\langle\mathbf{a}\rangle$ when $f\langle\mathbf{a}\rangle$ is of real type $t$.

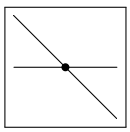
For each real $d$-type $t$ we also sketch a simple picture that shows the relevant part of the graph of the function $y = f\langle\mathbf{a}\rangle$, where $f\langle\mathbf{a}\rangle \in \mathbb{R}[x]$ is of degree $d$ and real type $t$. On all our pictures we draw the $x$-axis as the horizontal axis. Of course, the actual slope, curvature, or even the multiplicity of the real roots of the graphed functions possibly depend on actual parameter values. Nevertheless, the sign and the relative position of the real roots remain unchanged for any parameter values $\mathbf{a}$ satisfying the respective guard $\gamma$.

Let $f = ax + b \in \mathbb{Z}[\mathbf{u}][x]$ be such that $\operatorname{coeffs} f \subset \mathbb{Z}[\mathbf{u}]$ and $\operatorname{lc} f \neq 0$. All the real 1-types along with their respective guards for $f$ are:

1. $(-1, 0, 1)$ with guard $\gamma$:
$$a > 0$$



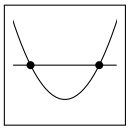
$-1$. $(1, 0, -1)$ with guard $\gamma$:
$$a < 0$$



Let $f = ax^2 + bx + c \in \mathbb{Z}[\mathbf{u}][x]$ be such that coeffs $f \subset \mathbb{Z}[\mathbf{u}]$ and $\operatorname{lc} f \neq 0$. All the real 2-types along with their respective guards for $f$ are:

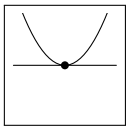
1. $(1, 0, -1, 0, 1)$ with guard $\gamma$:

$$a > 0 \wedge b^2 - 4ac > 0$$



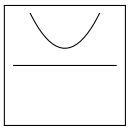
2. $(1, 0, 1)$ with guard $\gamma$:

$$a > 0 \wedge b^2 - 4ac = 0$$



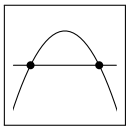
3. $(1)$ with guard $\gamma$:

$$a > 0 \wedge b^2 - 4ac < 0$$



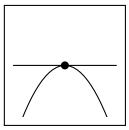
$-1$. $(-1, 0, 1, 0, -1)$ with guard $\gamma$:

$$a < 0 \wedge b^2 - 4ac > 0$$



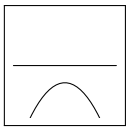
$-2$. $(-1, 0, -1)$ with guard $\gamma$:

$$a < 0 \wedge b^2 - 4ac = 0$$



$-3$. $(-1)$ with guard $\gamma$:

$$a < 0 \wedge b^2 - 4ac < 0$$



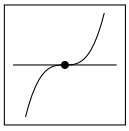
Let $f = ax^3 + bx^2 + cx + d \in \mathbb{Z}[\mathbf{u}][x]$ be such that coeffs $f \subset \mathbb{Z}[\mathbf{u}]$ and $\operatorname{lc} f \neq 0$. Denote by $D_f$ the discriminant of $f$:

$$D_f = -b^2c^2 + 4c^3a + 4b^3d + 27d^2a^2 - 18abcd.$$

All the real 3-types along with their respective guards for $f$ are:

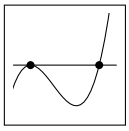
1. $(-1, 0, 1)$ with guard $\gamma$:

$$a > 0 \wedge (-b^2 + 3ac \geq 0 \vee D_f > 0)$$



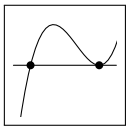
2. $(-1, 0, -1, 0, 1)$ with guard $\gamma$:

$$a > 0 \wedge -b^2 + 3ac < 0 \wedge D_f = 0 \wedge 2b^3 + 27da^2 - 9abc < 0$$



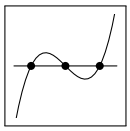
3. $(-1, 0, 1, 0, 1)$ with guard $\gamma$:

$$a > 0 \wedge -b^2 + 3ac < 0 \wedge D_f = 0 \wedge 2b^3 + 27da^2 - 9abc > 0$$



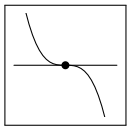
4. $(-1, 0, 1, 0, -1, 0, 1)$ with guard $\gamma$:

$$a > 0 \wedge -b^2 + 3ac < 0 \wedge D_f < 0$$



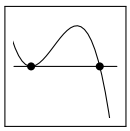
$-1$. $(1, 0, -1)$ with guard $\gamma$:

$$a < 0 \wedge (-b^2 + 3ac \geq 0 \vee D_f > 0)$$



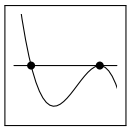
$-2$. $(1, 0, 1, 0, -1)$ with guard $\gamma$:

$$a < 0 \wedge -b^2 + 3ac < 0 \wedge D_f = 0 \wedge 2b^3 + 27da^2 - 9abc > 0$$



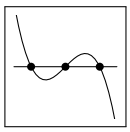
$-3$. $(1, 0, -1, 0, -1)$ with guard $\gamma$:

$$a < 0 \wedge -b^2 + 3ac < 0 \wedge D_f = 0 \wedge 2b^3 + 27da^2 - 9abc < 0$$



$-4$. $(1, 0, -1, 0, 1, 0, -1)$ with guard $\gamma$:

$$a < 0 \wedge -b^2 + 3ac < 0 \wedge D_f < 0$$



From now on we will refer to a real $d$-type, $d \in \{1, 2, 3\}$, by its respective number given in the real type lists given above instead of its full sequence of signs. E.g., the real 3-type $(1, 0, -1)$ is simply "real 3-type $-1$," and the real 2-type $(1, 0, -1, 0, 1)$ is simply "real 2-type 1."

The completeness of these real type lists and the correctness of their respective guards follow directly from the result of Weispfenning [83, Corollary 2.2]. We rephrase this result here in our setting and notation:

**Theorem 39** (Real Types and Guards up to Degree Three)**.** *Let $f$ be a polynomial from $\mathbb{Z}[\mathbf{u}][x]$ of degree $d \in \{1, 2, 3\}$. Then the following hold:*

*(i) Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values such that $\deg f\langle \mathbf{a} \rangle = d$. Then $f\langle \mathbf{a} \rangle$ is of exactly one of the real $d$-types listed above.*

*(ii) Let $\mathbf{a} \in \mathbb{R}^m$ be arbitrary parameter values. The polynomial $f\langle \mathbf{a} \rangle$ is of some listed real $d$-type $t$ if and only if $\mathbf{a}$ satisfies the given guard $\gamma$ of $t$.* □

Now we are ready to describe the three crucial algorithms. We begin with algorithm `at-cs` that computes a set of candidate solutions for a given atomic formula $f \varrho 0$:

Algorithm `at-cs`$(f \varrho 0, x)$.
Input: an atomic formula $f \varrho 0$, where $f \in \mathbb{Z}[\mathbf{u}][x]$, is a polynomial of degree $d$ and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$.
Output: a set $\mathfrak{c}$ of candidate solutions for $f \varrho 0$.

1. If $d = 0$, then return $\emptyset$.

2. $\mathfrak{c} := \emptyset$

3. If $(\operatorname{lc} f)\langle \mathbf{a} \rangle = 0$ for some $\mathbf{a} \in \mathbb{R}^m$, then

   3.1. $\mathfrak{c} := \texttt{at-cs}(\operatorname{red} f\ \varrho\ 0, x)$

4. If $\varrho$ is "$=$," then

   4.1. For each real $d$-type $t = (s_1, \ldots, s_{2k+1})$ do

      4.1.1. For each $i \in \{1, \ldots, k\}$ add $(f, (t, i), \mathrm{IP})$ to $\mathfrak{c}$.

5. If $\varrho$ is "$\neq$," then

   5.1. For each real $d$-type $t = (s_1, \ldots, s_{2k+1})$ do

      5.1.1. For each $i \in \{1, \ldots, k\}$ add $(f, (t, i), \mathrm{EP})$ to $\mathfrak{c}$.

6. If $\varrho$ is "$<$," then

   6.1. For each real $d$-type $t = (s_1, \ldots, s_{2k+1})$ do

      6.1.1. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = -1$ and $s_{2i+1} = 1$ add $(f, (t, i), \mathrm{SUB})$ to $\mathfrak{c}$.

      6.1.2. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = 1$ and $s_{2i+1} = -1$ add $(f, (t, i), \mathrm{SLB})$ to $\mathfrak{c}$.

      6.1.3. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = -1$ and $s_{2i+1} = -1$ add $(f, (t, i), \mathrm{EP})$ to $\mathfrak{c}$.

7. If $\varrho$ is "$\leq$," then

   7.1. For each real $d$-type $t = (s_1, \ldots, s_{2k+1})$ do

      7.1.1. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = -1$ and $s_{2i+1} = 1$ add $(f, (t, i), \mathrm{WUB})$ to $\mathfrak{c}$.

      7.1.2. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = 1$ and $s_{2i+1} = -1$ add $(f, (t, i), \mathrm{WLB})$ to $\mathfrak{c}$.

      7.1.3. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = 1$ and $s_{2i+1} = 1$ add $(f, (t, i), \mathrm{IP})$ to $\mathfrak{c}$.

8. If $\varrho$ is "$\geq$," then

   8.1. For each real $d$-type $t = (s_1, \ldots, s_{2k+1})$ do

      8.1.1. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = 1$ and $s_{2i+1} = -1$ add $(f, (t, i), \mathrm{WUB})$ to $\mathfrak{c}$.

      8.1.2. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = -1$ and $s_{2i+1} = 1$ add $(f, (t, i), \mathrm{WLB})$ to $\mathfrak{c}$.

      8.1.3. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = -1$ and $s_{2i+1} = -1$ add $(f, (t, i), \mathrm{IP})$ to $\mathfrak{c}$.

9. If $\varrho$ is "$>$," then

   9.1. For each real $d$-type $t = (s_1, \ldots, s_{2k+1})$ do

      9.1.1. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = -1$ and $s_{2i+1} = 1$ add $(f, (t, i), \mathrm{SLB})$ to $\mathfrak{c}$.

      9.1.2. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = 1$ and $s_{2i+1} = -1$ add $(f, (t, i), \mathrm{SUB})$ to $\mathfrak{c}$.

9.1.3. For each $i \in \{1, \ldots, k\}$ such that $s_{2i-1} = 1$ and $s_{2i+1} = 1$ add $(f, (t, i), \mathrm{EP})$ to $\mathfrak{c}$.

10. Return $\mathfrak{c}$.

Before we go on we make a few comments on algorithm `at-cs`. First, observe that the algorithm does not generate any candidate solutions for the cases when $f$ has no real roots. For example, no candidate solution with real 2-type 3 or $-3$ will ever be generated. Second, the question whether the leading coefficient of $f$ can vanish—which we ask in step 3—is an optimization that can be left out. If it is too hard to decide whether the leading coefficient can vanish, then it is sufficient to simply include the candidate solutions $\texttt{at-cs}(\mathrm{red}\, f\ \varrho\ 0, x)$ in $\mathfrak{c}$. Third, it is noteworthy that the algorithm is not limited to degree three; it can be used for any $d$-degree polynomial as soon as we provide it with all real $d$-types.

We continue with algorithm `guard`, which simply uses the guards for $f$ listed above. Similarly as with `at-cs`, it is obvious that this algorithm works for a higher degree $d$ as soon as we deliver a guard $\gamma_t$ for each real $d$-type $t$:

Algorithm $\texttt{guard}\big((f, S), x\big)$.
Input: a parametric root description $(f, S)$, where $f \in \mathbb{Z}[\mathbf{u}][x]$ with $\deg f \leq 3$ and $S = \{(t_1, r_1), \ldots, (t_k, r_k)\}$.
Output: a quantifier-free Tarski formula $\gamma$, which is a guard of $(f, S)$.

1. Let $T = \{t_i\}_{i=1}^{k}$ be the set of all real $d$-types occurring in $S$.

2. $\gamma := \bigvee_{t \in T} \gamma_t$, where $\gamma_t$ is the guard of real $d$-type $t$ for $f$ listed above

3. Return $\gamma$.

Indeed the most challenging part is to give algorithm `vs-prd-at`, which actually "does the work." We first explain its main idea on two example situations.

As the first example consider the situation when we want to substitute parametric root description $\lambda = (f, (1, 1))$, where $f = x^2 + bx - 7$ and $b \in \mathbb{Z}[\mathbf{u}]$, into atomic formula $g \geq 0$, where $g = a^*x + b^*$ and $a^*$, $b^* \in \mathbb{Z}[\mathbf{u}]$. We want necessary and sufficient conditions in the parameters $\mathbf{u}$ for $g$ to be non-negative at the first root of $f$ under the assumption that $f$ is of real 2-type 1, i.e., we want a Tarski formula equivalent to $(g \geq 0)[x /\!\!/ \lambda]$. Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values such that $f\langle \mathbf{a} \rangle$ is of real 2-type 1. Denote by $\alpha$ the first real root of $f\langle \mathbf{a} \rangle$. We distinguish the following three cases:

1. If $a^*\langle \mathbf{a} \rangle$ is positive, then $g\langle \mathbf{a} \rangle(\alpha)$ is non-negative if and only if the root $\beta = -\frac{b^*\langle \mathbf{a} \rangle}{a^*\langle \mathbf{a} \rangle}$ of $g\langle \mathbf{a} \rangle$ is less than or equal to $\alpha$, i.e., $\beta$ is identical with or lies to the left of $\alpha$.

2. If $a^*\langle \mathbf{a} \rangle$ is negative, then $g\langle \mathbf{a} \rangle(\alpha)$ is non-negative if and only if the root $\beta = -\frac{b^*\langle \mathbf{a} \rangle}{a^*\langle \mathbf{a} \rangle}$ of $g\langle \mathbf{a} \rangle$ is greater than or equal to $\alpha$, i.e., $\beta$ is identical with or lies to the right of $\alpha$.

3. If $a^*\langle \mathbf{a} \rangle$ is zero, then $g\langle \mathbf{a} \rangle(\alpha)$ is non-negative if and only if $b^*\langle \mathbf{a} \rangle(\alpha) = b^*\langle \mathbf{a} \rangle$ is non-negative.

The conditions "$\beta$ lies to the left of $\alpha$" and "$\beta$ lies to the right of $\alpha$" can be expressed by sign constraints on $f\langle\mathbf{a}\rangle$ at the root $\beta = -\frac{b^*\langle\mathbf{a}\rangle}{a^*\langle\mathbf{a}\rangle}$ of $g\langle\mathbf{a}\rangle$ and $g\langle\mathbf{a}\rangle$ at the root $-\frac{b\langle\mathbf{a}\rangle}{2}$ of $f'\langle\mathbf{a}\rangle$. Using these observations we obtain equivalent conditions in the parameters $\mathbf{u}$:

1. $a^* > 0 \wedge f(-\frac{b^*}{a^*}) \geq 0 \wedge g(-\frac{b}{2}) > 0$,
2. $a^* < 0 \wedge \big(f(-\frac{b^*}{a^*}) \leq 0 \vee g(-\frac{b}{2}) > 0\big)$,
3. $a^* = 0 \wedge b^* \geq 0$,

where $f(-\frac{b^*}{a^*})$ is obtained by replacing in $f$ each occurrence of $x$ with $-\frac{b^*}{a^*}$, and $g(-\frac{b}{2})$ is obtained by replacing in $g$ each occurrence of $x$ with $-\frac{b}{2}$. Even though this is not a Tarski formula, it is not hard to see that this semantically makes sense. Observe that this can, in fact, be expressed by linear virtual substitution as follows:

1. $a^* > 0 \wedge (f \geq 0)[x \mathbin{/\!\!/} (a^*x + b^*, (1,1))] \wedge (g > 0)[x \mathbin{/\!\!/} (f', (1,1))]$,
2. $a^* < 0 \wedge \big((f \leq 0)[x \mathbin{/\!\!/} (a^*x + b^*, (-1,1))] \vee (g > 0)[x \mathbin{/\!\!/} (f', (1,1))]\big)$,
3. $a^* = 0 \wedge b^* \geq 0$.

Carrying out the linear virtual substitutions using algorithms `pseudo-sgn-rem` and `vs-prd-at-1` and forming a disjunction of the three formulas, we finally obtain a Tarski formula equivalent to $(g \geq 0)[x \mathbin{/\!\!/} \lambda]$:

$$\begin{aligned}
&a^* > 0 \wedge -7{a^*}^2 - ba^*b^* + {b^*}^2 \geq 0 \wedge -a^*b + 2b^* > 0 \vee \\
&a^* < 0 \wedge \big(-7{a^*}^2 - ba^*b^* + {b^*}^2 \leq 0 \vee -a^*b + 2b^* > 0\big) \vee \\
&a^* = 0 \wedge b^* \geq 0.
\end{aligned}$$

As the second example imagine that we want to substitute a parametric root description $\lambda = (f, (4,1))$, where $f \in \mathbb{Z}[\mathbf{u}][x]$ is a polynomial of degree 3, into an atomic formula $g < 0$, where $g \in \mathbb{Z}[\mathbf{u}][x]$ is a polynomial of degree 2. Assume that we are given parameter values $\mathbf{a} \in \mathbb{R}^m$ such that $f\langle\mathbf{a}\rangle$ is of real 3-type 4 and $g\langle\mathbf{a}\rangle$ is of real 2-type 1. Let $\beta_1 = (g, (1,1))$ and $\beta_2 = (g, (1,2))$ be parametric root descriptions describing the first and the second real root of $g$, respectively. Then we have $\mathbb{R} \models (g < 0)(\mathbf{a}, \lambda\langle\mathbf{a}\rangle)$ if and only if $\beta_1\langle\mathbf{a}\rangle$ lies to the left of the point $\lambda\langle\mathbf{a}\rangle$ and $\beta_2\langle\mathbf{a}\rangle$ lies to the right of the point $\lambda\langle\mathbf{a}\rangle$.

In Figure 2.3 we depict two possible situations what the relative positions of $\lambda\langle\mathbf{a}\rangle$, $\beta_1\langle\mathbf{a}\rangle$, and $\beta_2\langle\mathbf{a}\rangle$ could look like. In the first case we obviously have $\mathbb{R} \models (g < 0)(\mathbf{a}, \lambda\langle\mathbf{a}\rangle)$ and in the second case we have $\mathbb{R} \not\models (g < 0)(\mathbf{a}, \lambda\langle\mathbf{a}\rangle)$. The pictures make clear that the "to-the-left and to-the-right of $\lambda\langle\mathbf{a}\rangle$" view indeed precisely characterizes those parameter values $\mathbf{a}$ such that $\mathbb{R} \models (g < 0)(\mathbf{a}, \lambda\langle\mathbf{a}\rangle)$ under the assumptions that $f\langle\mathbf{a}\rangle$ is of real 3-type 4 and $g\langle\mathbf{a}\rangle$ is of real 2-type 1.

Our aim now is to obtain necessary and sufficient conditions in the parameters $\mathbf{u}$ for the statement "$\beta_1\langle\mathbf{a}\rangle$ is to the left of $\lambda\langle\mathbf{a}\rangle$." We achieve this by carefully analyzing the possible signs and shapes of the derivatives of $f\langle\mathbf{a}\rangle$ relative to $g\langle\mathbf{a}\rangle$. Figure 2.2 shows $f\langle\mathbf{a}\rangle$ together with its derivatives. Notice that the curvature or even the relative position of the first root of $f''\langle\mathbf{a}\rangle$ and the second root of $f\langle\mathbf{a}\rangle$ possibly depend on parameter values $\mathbf{a}$. The real types of $f'\langle\mathbf{a}\rangle$ and $f''\langle\mathbf{a}\rangle$, in contrast, remain in this case the same for any parameter values

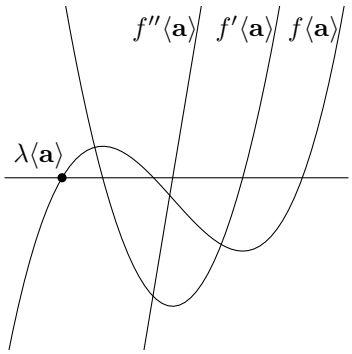



Figure 2.2: A polynomial $f\langle\mathbf{a}\rangle$ of real 3-type 4 with its derivatives and its root $\lambda\langle\mathbf{a}\rangle = (f, (4, 1))\langle\mathbf{a}\rangle$.

$\mathbf{a}$ such that $f\langle\mathbf{a}\rangle$ is of real 3-type 4. Therefore, we can deduce that "$\beta_1\langle\mathbf{a}\rangle$ is to the left of $\lambda\langle\mathbf{a}\rangle$" if and only if $f\langle\mathbf{a}\rangle$ is negative at $\beta_1\langle\mathbf{a}\rangle$, $f'\langle\mathbf{a}\rangle$ is positive at $\beta_1\langle\mathbf{a}\rangle$, and $f''\langle\mathbf{a}\rangle$ is negative at $\beta_1\langle\mathbf{a}\rangle$, because we assume that $f\langle\mathbf{a}\rangle$ is of real 3-type 4 and $g\langle\mathbf{a}\rangle$ is of real 2-type 1. Under these assumptions, the definition of virtual substitution ensures that "$\beta_1\langle\mathbf{a}\rangle$ is to the left of $\lambda\langle\mathbf{a}\rangle$" if and only if

$$\mathbb{R} \models \big((f < 0)[x /\!\!/ \beta_1] \wedge (f' > 0)[x /\!\!/ \beta_1] \wedge (f'' < 0)[x /\!\!/ \beta_1]\big)(\mathbf{a}).$$

Proceeding in a similar fashion, we obtain a necessary and sufficient condition in the parameters $\mathbf{u}$ for the statement "$\beta_2\langle\mathbf{a}\rangle$ is to the right of $\lambda\langle\mathbf{a}\rangle$:" Observe that this is the case if and only if $f\langle\mathbf{a}\rangle$ is positive at point $\beta_2\langle\mathbf{a}\rangle$, or $g\langle\mathbf{a}\rangle$ is smaller than zero at point $\alpha_1\langle\mathbf{a}\rangle$, where $\alpha_1 = (f', (1, 1))$, i.e., $\alpha_1$ describes the first root of the first derivative of $f$. Since $g\langle\mathbf{a}\rangle$ is of real 2-type 1, the latter case covers the case when $\beta_2\langle\mathbf{a}\rangle$ is to the right of the first root of $f\langle\mathbf{a}\rangle$, but $f\langle\mathbf{a}\rangle$ is non-positive at $\beta_2\langle\mathbf{a}\rangle$. The definition of virtual substitution therefore ensures that "$\beta_2\langle\mathbf{a}\rangle$ is to the right of $\lambda\langle\mathbf{a}\rangle$" if and only if

$$\mathbb{R} \models \big((f > 0)[x /\!\!/ \beta_2] \vee (g < 0)[x /\!\!/ \alpha_1]\big)(\mathbf{a}).$$

Putting these together, we have just proven the following for any parameter values $\mathbf{a} \in \mathbb{R}$: If $f\langle\mathbf{a}\rangle$ is of real 3-type 4 and $g\langle\mathbf{a}\rangle$ is of real 2-type 1, then $(g < 0)[x /\!\!/ (f, (4, 1))]$ holds if and only if $\mathbb{R} \models \psi(\mathbf{a})$ holds, where

$$\begin{aligned}\psi : \big((f < 0)[x /\!\!/ \beta_1] \wedge (f' > 0)[x /\!\!/ \beta_1] \wedge (f'' < 0)[x /\!\!/ \beta_1]\big) \wedge \\ \big((f > 0)[x /\!\!/ \beta_2] \vee (g < 0)[x /\!\!/ \alpha_1]\big).\end{aligned}$$

Observe that all the virtual substitutions occurring in $\psi$ are quadratic and linear virtual substitutions, so they can be realized by algorithms for quadratic and linear virtual substitution, respectively.

Analyzing all relevant situations in a similar fashion, we derive all finitely many formula schemes that yield necessary and sufficient conditions in the parameters $\mathbf{u}$ for $g \varrho 0$ to hold at a root of $f$ for any cubic polynomial $f$ and a linear or quadratic polynomial $g$. We systematically list these formula schemes along with formula schemes for the quadratic virtual substitution in Appendix A.1 and Appendix A.2. Each formula presented there is a simplified equivalent of

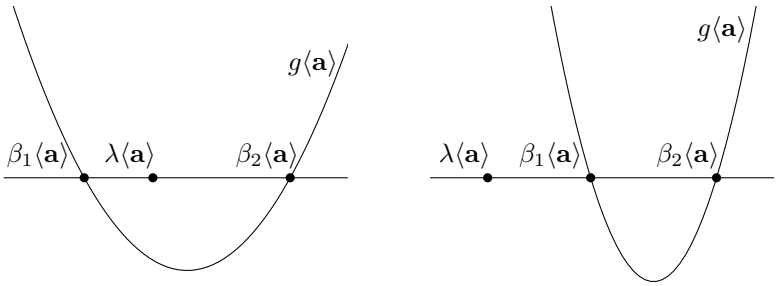



Figure 2.3: Two different potential situations of position $\lambda\langle\mathbf{a}\rangle$ relative to $g\langle\mathbf{a}\rangle$: On the left picture the formula $g < 0$ is satisfied by $(\mathbf{a}, \lambda\langle\mathbf{a}\rangle)$, because $\beta_1\langle\mathbf{a}\rangle$ is to the left and $\beta_2\langle\mathbf{a}\rangle$ to the right of $\lambda\langle\mathbf{a}\rangle$. On the right picture the formula $g < 0$ is not satisfied by $(\mathbf{a}, \lambda\langle\mathbf{a}\rangle)$, because $\beta_1\langle\mathbf{a}\rangle$ is not to the left of $\lambda\langle\mathbf{a}\rangle$.

a formula obtained by an analysis of one relevant situation. For concrete polynomials $f$ and $g$, a necessary and sufficient condition for $(g\ \varrho\ 0)[x /\!\!/ (f, S)]$ to hold is obtained by substituting the coefficients of $f$ and $g$ into the appropriate scheme. With these formula schemes at hand, it is straightforward to give algorithms `vs-prd-at-2` and `vs-prd-at-3`:

Algorithm `vs-prd-at-2`$\big(g\ \varrho\ 0, (f, S), x\big)$:
Input: an atomic formula $g\ \varrho\ 0$ and a parametric root description $(f, S)$ such that $g$, $f \in \mathbb{Z}[\mathbf{u}][x]$, $\varrho \in \{=, \neq, <, \leq, \geq, >\}$, and $\deg g < \deg f = 2$.
Output: $(g\ \varrho\ 0)[x /\!\!/ (f, S)]$, i.e., a quantifier-free formula in the parameters $\mathbf{u}$ meeting our specification of virtual substitution.

1. If $\deg g = 0$, then return $g\ \varrho\ 0$.
2. If there is no formula scheme in Appendix A.1 involving the set $S$, then
   2.1. $(f, S) := (-f, -S)$
3. If $\varrho \in \{=, <, \leq\}$, then
   3.1. $\psi := (g\ \varrho\ 0)[x /\!\!/ (f, S)]$
   Use the appropriate formula scheme for $(f, S)$ from Appendix A.1.
4. If $\varrho \in \{\neq, \geq, >\}$, then
   4.1. $\psi := \big(\neg(g\ \varrho\ 0)\big)[x /\!\!/ (f, S)]$
   Use the appropriate formula scheme for $(f, S)$ from Appendix A.1.
   4.2. $\psi := \neg\psi$
5. Return $\psi$.

Algorithm `vs-prd-at-3`$\big(g\ \varrho\ 0, (f, S), x\big)$:
Input: an atomic formula $g\ \varrho\ 0$ and a parametric root description $(f, S)$ such that $g$, $f \in \mathbb{Z}[\mathbf{u}][x]$, $\varrho \in \{=, \neq, <, \leq, \geq, >\}$, and $\deg g < \deg f = 3$.
Output: $(g\ \varrho\ 0)[x /\!\!/ (f, S)]$, i.e., a quantifier-free formula in the parameters $\mathbf{u}$ meeting our specification of virtual substitution.

1. If $\deg g = 0$, then return $g\ \varrho\ 0$.

2. If there is no formula scheme in Appendix A.2 involving the set $S$, then

   2.1. $(f, S) := (-f, -S)$

3. If $\varrho \in \{=, <, \leq\}$, then

   3.1. $\psi_+ := (g\ \varrho\ 0))[x /\!\!/ (f, S)]$
   $\psi_- := (-g\ \varrho\ 0)[x /\!\!/ (f, S)]$
   $\psi_0 := \texttt{vs-prd-at-3}\big(\operatorname{red} g\ \varrho\ 0, (f, S), x\big)$
   Use the appropriate formula scheme for $(f, S)$ from Appendix A.2.

4. If $\varrho \in \{=, <, \leq\}$, then

   4.1. $\psi_+ := \big(\neg(g\ \varrho\ 0)\big)[x /\!\!/ (f, S)]$
   $\psi_- := \big(\neg(-g\ \varrho\ 0)\big)[x /\!\!/ (f, S)]$
   $\psi_0 := \texttt{vs-prd-at-3}\big(\neg(\operatorname{red} g\ \varrho\ 0), (f, S), x\big)$
   Use the appropriate formula scheme for $(f, S)$ from Appendix A.2.

   4.2. $\psi_+ := \neg\psi_+$; $\psi_- := \neg\psi_-$; $\psi_0 := \neg\psi_0$

5. Return $\operatorname{lc} g > 0 \wedge \psi_+ \vee \operatorname{lc} g < 0 \wedge \psi_- \vee \operatorname{lc} g = 0 \wedge \psi_0$.

For a set of root specifications $S = \{(t_1, r_1), \ldots, (t_k, r_k)\}$ we define $-S$ to be the set $\{(-t_1, r_1), \ldots, (-t_k, r_k)\}$. To justify the correctness of algorithms `vs-prd-at-2` and `vs-prd-at-3` we will use the following two lemmas stating that $(f, S)$ and $(-f, -S)$ are equivalent in the following sense: They have equivalent guards, and they yield equivalent formulas when substituted by means of virtual substitution. It is therefore correct to use a formula scheme for $(-f, -S)$ when we want to realize the virtual substitution of $(f, S)$ and vice versa.

**Lemma 40.** *Let $f \in \mathbb{Z}[\mathbf{u}][x]$. Let $(f, S)$, where $S = \{(t_1, r_1), \ldots, (t_k, r_k)\}$, be a parametric root description of $f$. Let $\gamma$ be a guard of $(f, S)$. Let $\gamma'$ be a guard of $(-f, -S)$. Then $\mathbb{R} \models \gamma \longleftrightarrow \gamma'$.*

*Proof.* Let $\mathbf{a} \in \mathbb{R}$ be some parameter values. We show that $\mathbf{a}$ satisfies $\gamma$ if and only if $\mathbf{a}$ satisfies $\gamma'$.

Assume first that $\mathbf{a}$ satisfies $\gamma$. Since $\gamma$ is a guard of $(f, S)$, the univariate polynomial $f\langle\mathbf{a}\rangle$ is of some real type $t_j$. This ensures that $-(f\langle\mathbf{a}\rangle)$ is of real type $-t_j$ for some $j \in \{1, \ldots, k\}$. At the same time we have $-(f\langle\mathbf{a}\rangle) = (-f)\langle\mathbf{a}\rangle$, because $\langle\cdot\rangle$ is a homomorphism. The assumption that $\gamma'$ is a guard of $(-f, -S)$ together with the fact that $(-f)\langle\mathbf{a}\rangle$ is of real type $-t_j$ imply that $\mathbf{a}$ satisfies $\gamma'$. This proves that $\mathbf{a}$ satisfies $\gamma'$ as well, i.e., $\mathbb{R} \models \gamma \longrightarrow \gamma'$.

Assume now that $\mathbf{a}$ satisfies $\gamma'$. Since $\gamma'$ is a guard of $(-f, -S)$, the univariate polynomial $(-f)\langle\mathbf{a}\rangle$ is of some real type $-t_j$. This ensures that $(-f)\langle\mathbf{a}\rangle = -(f\langle\mathbf{a}\rangle)$ is of real type $t_j$, $j \in \{1, \ldots, k\}$. Since we assume that $\gamma$ is a guard of $(f, S)$, we obtain that $\mathbf{a}$ satisfies $\gamma$. This shows that $\mathbb{R} \models \gamma' \longrightarrow \gamma$, so the proof is finished. □

**Lemma 41.** *Let $g\ \varrho\ 0$, where $g \in \mathbb{Z}[\mathbf{u}][x]$ and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$, be an atomic formula. Let $f \in \mathbb{Z}[\mathbf{u}][x]$, and let $(f, S)$, where the set of root specifications $S$ is $\{(t_1, r_1), \ldots, (t_k, r_k)\}$, be a parametric root description of $f$. Let $\gamma$ be a guard of $(f, S)$. Then*

$$\mathbb{R} \models \gamma \longrightarrow \big((g\ \varrho\ 0)[x /\!\!/ (f, S)] \longleftrightarrow (g\ \varrho\ 0)[x /\!\!/ (-f, -S)]\big).$$

*Proof.* Let $\mathbf{a} \in \mathbb{R}^m$ be some parameter values. Assume that $\mathbf{a}$ satisfies $\gamma$. We need to prove that $\mathbf{a}$ satisfies the formula $(g \ \varrho \ 0)[x \mathbin{/\!\!/} (f, S)] \longleftrightarrow (g \ \varrho \ 0)[x \mathbin{/\!\!/} (-f, -S)]$ as well.

Let us first assume that $\mathbf{a}$ satisfies $(g \ \varrho \ 0)[x \mathbin{/\!\!/} (f, S)]$. Since we assume that $\mathbf{a}$ satisfies $\gamma$, this means that there exists $j \in \{1, \ldots, k\}$ such that $f\langle \mathbf{a} \rangle$ is of real type $t_j$, and $g \ \varrho \ 0$ is satisfied by the $r_j$-th real root of $f\langle \mathbf{a} \rangle$. We show that $\mathbf{a}$ satisfies $(g \ \varrho \ 0)[x \mathbin{/\!\!/} (-f, -S)]$ as well. Observe that $-(f\langle \mathbf{a} \rangle) = (-f)\langle \mathbf{a} \rangle$ is of real type $-t_j$. This ensures that the $r_j$-th real root of $(-f)\langle \mathbf{a} \rangle$ and the $r_j$-th real root of $f\langle \mathbf{a} \rangle$ are actually the same real number. By assumption, $g \ \varrho \ 0$ holds at this root. On the other hand, Lemma 40 ensures that $\gamma$ is a guard of $(-f, -S)$. Therefore, the definition of a virtual substitution ensures that the statement that $g \ \varrho \ 0$ holds at the $r_j$-th root of $(-f)\langle \mathbf{a} \rangle$ is equivalent to the statement that $\mathbf{a}$ satisfies $(g \ \varrho \ 0)[x \mathbin{/\!\!/} (-f, -S)]$. This proves that $\mathbf{a}$ satisfies $(g \ \varrho \ 0)[x \mathbin{/\!\!/} (-f, -S)]$ as well.

The proof of the converse implication is similar, so we omit it. □

Observe that the formula schemes of Section A.1 and Section A.2 cover only the cases when $\varrho \in \{=, <, \leq\}$ and the real type of $f$ is "positive," i.e., $\operatorname{lc} f$ is greater than zero. Negating appropriately and using Lemmas 40 and 41 it is easy to see that all the remaining cases can be converted to these cases. We exploit this in `vs-prd-at-2` and `vs-prd-at-3`.

We are using these symmetries to reduce the number of cases for which we need to derive a formula scheme. Notice that using these symmetries does not enlarge the resulting formulas, because we do not rewrite "$\leq$" as "$=$ or $<$." Even though this would have reduced the number of cases, the size of the formulas $(g \leq 0)[x \mathbin{/\!\!/} (f, S)]$ would grow. Therefore we explicitly give formula schemes also for the cases when $\varrho$ is "$\leq$."

**Theorem 42.** *Algorithms* `at-cs`*,* `guard`*,* `vs-prd-2`*, and* `vs-prd-3` *meet their specifications.*

*Proof.* To prove the correctness of algorithm `at-cs`, we first fix values $\mathbf{a} \in \mathbb{R}^m$ for the parameters $\mathbf{u}$, and proceed by the induction on $d$.

If $d = 0$, then $\Phi(f \ \varrho \ 0, \mathbf{a})$ is either $\emptyset$ or $\mathbb{R}$, so there is no boundary point to cover. Consequently, $\emptyset$ is a set of candidate solutions.

Assume that $d > 0$, and that `at-cs` meets its specification for polynomials of degree smaller than $d$. If $(\operatorname{lc} f)\langle \mathbf{a} \rangle = 0$, then $\mathfrak{c}$ is the satisfying set for $\operatorname{red} f \ \varrho \ 0$ by the induction hypothesis, so each boundary point of $\Phi(f \ \varrho \ 0, \mathbf{a})$ is properly covered. If $(\operatorname{lc} f)\langle \mathbf{a} \rangle \neq 0$, then $\Phi(f \ \varrho \ 0, \mathbf{a})$ is a finite union of intervals, and $f\langle \mathbf{a} \rangle$ is of some real $d$-type $(s_1, \ldots, s_{2k+1})$. If $\varrho$ is "$=$," then $\Phi(f \ \varrho \ 0, \mathbf{a})$ consists of $k$ isolated points, which are obviously all properly covered by $\mathfrak{c}$. Assume that $\varrho$ is "$<$," and consider the $i$-th root of $f\langle \mathbf{a} \rangle$. There are four possibilities what $f\langle \mathbf{a} \rangle$ looks like near each of this root:

1. $s_{2i-1} = -1$ and $s_{2i+1} = 1$: The $i$-th root is a strict upper bound boundary point of $\Phi(f \ \varrho \ 0, \mathbf{a})$.

2. $s_{2i-1} = 1$ and $s_{2i+1} = -1$: The $i$-th root is a strict lower bound boundary point of $\Phi(f \ \varrho \ 0, \mathbf{a})$.

3. $s_{2i-1} = -1$ and $s_{2i+1} = -1$: The $i$-th root is an excluded point boundary point of $\Phi(f \ \varrho \ 0, \mathbf{a})$.

4. $s_{2i-1} = 1$ and $s_{2i+1} = 1$: The $i$-th root is not a boundary point.

Looking at the pseudocode of `vs-at`, we see that all boundary points are properly covered by `c`. This finishes the proof for the case when $\varrho$ is "$<$." The proof is similar for other relations, so we omit it. This proves that `at-cs` meets its specification.

The correctness of algorithm `guard` follows directly from Theorem 39.

The correctness of Algorithms `vs-prd-2` and `vs-prd-3` follows from the correctness of the formula schemes given in Appendix A.1 and Appendix A.2 combined with Lemmas 40 and 41. In this subsection we have shown on an example how to derive and prove correct one concrete formula scheme. We did a similar analysis and proof for each formula scheme given in the appendix. This finishes the proof of the theorem. □

To conclude this subsection let us note that the idea to consider all the possible relative positions of $f$ and $g$ for the cubic case and to "draw pictures" appeared already in [83]. Here we made this idea explicit within our framework and formalism.

### 2.5.3 Clustering

Looking closer at the instantiations of `vs-scheme` described in the previous subsections we observe the following: The set $S$ of any parametric root description $(f, S)$ consists of exactly one root specification. Consequently, each parametric root description substituted into a formula $\varphi$ by means of virtual substitution schemes of Appendix A.1 and Appendix A.2 contains exactly one root specification as well. Here we propose virtual substitution with *clustering* that merges two compatible parametric root descriptions $(f, S_1)$ and $(f, S_2)$ into $(f, S_1 \cup S_2)$, so that they can be substituted into $\varphi$ simultaneously in one go.

We first illustrate the core idea of clustering on a linear atomic formula. Assume that an atomic formula $f = 0$, where $f$ is $ax + b$ and $a, b \in \mathbb{Z}[\mathbf{u}]$, occurs in $\varphi$. According to our instantiation described in Subsection 2.5.1, `at-cs-1` adds candidate solutions $\big(f, (1,1), \mathrm{IP}\big)$ and $\big(f, (-1,1), \mathrm{IP}\big)$ to a candidate solution set for $\varphi$. This means that the instantiation of `vs-scheme` eventually computes the following two formulas:

$$a > 0 \wedge \varphi[x /\!\!/ (f, (1,1))] \quad \text{and} \quad a < 0 \wedge \varphi[x /\!\!/ (f, (-1,1))].$$

The idea of clustering is to compute only one formula

$$a \neq 0 \wedge \varphi[x /\!\!/ (f, \{(1,1), (-1,1)\})]$$

instead. This essentially means that we substitute $(f, (1,1))$ and $(f, (-1,1))$ simultaneously and replace

$$a > 0 \wedge \varphi[x /\!\!/ (f, (1,1))] \vee a < 0 \wedge \varphi[x /\!\!/ (f, (-1,1))]$$

with $a \neq 0 \wedge \varphi[x /\!\!/ (f, \{(1,1), (-1,1)\})]$ in the quantifier-free equivalent of $\exists x(\varphi)$ computed by the instantiation of `vs-scheme`. Here note that $a \neq 0$ is a guard of $(f, \{(1,1), (-1,1)\})$.

To illustrate the difference between formulas obtained by simple and clustered virtual substitution we consider parametric root descriptions $(f, (1,1))$

and $(f, (-1, 1))$, where $f = ax + 1$. Imagine we want to substitute them into atomic formula $bx + c \geq 0$. Using algorithms `pseudo-sgn-rem` and `vs-at-1` we obtain $a^2c - ab \geq 0$ in both cases. Since in the first case we have a guard $a > 0$ and in the second case we have a guard $a < 0$, the obtained formula can be simplified to $ac - b \geq 0$ and $ac - b \leq 0$, respectively. Using clustering, we merge the two parametric root descriptions into $(f, \{(1, 1), (-1, 1)\})$. It is obvious that `vs-at-1` meets its specification also when given the parametric root description $(f, \{(1, 1), (-1, 1)\})$ as an argument. Therefore, algorithms `pseudo-sgn-rem` and `vs-at-1` ensure that $(bx + c \geq 0)[x /\!\!/ (f, \{(1, 1), (-1, 1)\})]$ is equivalent to $a^2c - ab \geq 0$. To sum up, using clustering we obtain formula $a^2c - ab \geq 0$ instead of two formulas $ac - b \geq 0$ and $ac - b \leq 0$ obtained by two simple (non-clustered) virtual substitutions followed by simplification.

Indeed, lifting the idea of clustering to higher degrees is more challenging, because we need to provide algorithms `vs-at-2` and `vs-at-3` that are able to handle clustered parametric root descriptions, i.e., parametric root descriptions containing more than one root specification. Before doing this, let us prove the following lemma that guarantees the correctness of clustering in general:

**Lemma 43** (Clustering Lemma)**.** *Let $f \in \mathbb{Z}[\mathbf{u}][x]$. Let $(f, S_1)$ and $(f, S_2)$ be two parametric root descriptions such that no real type occurs in both $S_1$ and $S_2$. Let $\gamma_1$ be a guard of $(f, S_1)$ and let $\gamma_2$ be a guard of $(f, S_2)$. Then $\gamma_1 \vee \gamma_2$ is a guard of the parametric root description $(f, S_1 \cup S_2)$, there exists a virtual substitution $[x /\!\!/ (f, S_1 \cup S_2)]$, and we have*

$$\mathbb{R} \models \gamma_1 \longrightarrow \big(\varphi[x /\!\!/ (f, S_1)] \longleftrightarrow \varphi[x /\!\!/ (f, S_1 \cup S_2)]\big), \quad \text{and}$$
$$\mathbb{R} \models \gamma_2 \longrightarrow \big(\varphi[x /\!\!/ (f, S_2)] \longleftrightarrow \varphi[x /\!\!/ (f, S_1 \cup S_2)]\big).$$

*Proof.* To begin with, observe that the assumption that $S_1$ and $S_2$ do not share a real type directly implies that $\gamma_1 \vee \gamma_2$ is a guard of $(f, S_1 \cup S_2)$. Furthermore, Proposition 24 ensures the existence of virtual substitution $[x /\!\!/ (f, S_1 \cup S_2)]$.

Let now $\mathbf{a} \in \mathbb{R}^m$ be arbitrary parameter values such that $\mathbb{R} \models \gamma_1(\mathbf{a})$ holds, i.e., $f\langle\mathbf{a}\rangle$ is of some real type $t$ occurring in $S_1$. We need to prove that $\mathbb{R} \models \varphi[x /\!\!/ (f, S_1)](\mathbf{a})$ if and only if $\mathbb{R} \models \varphi[x /\!\!/ (f, S_1 \cup S_2)](\mathbf{a})$.

First assume that $\mathbf{a}$ satisfies $\varphi[x /\!\!/ (f, S_1)]$. The definition of virtual substitution together with the assumption $\mathbb{R} \models \gamma_1(\mathbf{a})$ ensures that $\mathbb{R} \models \varphi(\mathbf{a}, (f, S_1)\langle\mathbf{a}\rangle)$. Since no real type occurs in both $S_1$ and $S_2$, we have $(f, S_1)\langle\mathbf{a}\rangle = (f, S_1 \cup S_2)\langle\mathbf{a}\rangle$, so $\mathbb{R} \models \varphi(\mathbf{a}, (f, S_1 \cup S_2)\langle\mathbf{a}\rangle)$. The definition of virtual substitution ensures that $\mathbb{R} \models \varphi[x /\!\!/ (f, S_1 \cup S_2)](\mathbf{a})$, because $\mathbf{a}$ obviously satisfies a guard of $(f, S_1 \cup S_2)$.

Now assume that $\mathbf{a}$ satisfies $\varphi[x /\!\!/ (f, S_1 \cup S_2)](\mathbf{a})$. Since we assume that $\mathbf{a}$ satisfies a guard of $(f, S_1 \cup S_2)$, the definition of virtual substitution implies that $\mathbb{R} \models \varphi(\mathbf{a}, (f, S_1 \cup S_2)\langle\mathbf{a}\rangle)$. Since no real type occurs in both $S_1$ and $S_2$, and $f\langle\mathbf{a}\rangle$ is of some real type occurring in $S_1$, we obtain $(f, S_1 \cup S_2)\langle\mathbf{a}\rangle = (f, S_1)\langle\mathbf{a}\rangle$, so in particular $\mathbb{R} \models \varphi(\mathbf{a}, (f, S_1)\langle\mathbf{a}\rangle)$. Finally, the definition of virtual substitution together with the assumption $\mathbb{R} \models \gamma_1(\mathbf{a})$ guarantee that $\mathbb{R} \models \varphi[x /\!\!/ (f, S_1)](\mathbf{a})$. This finishes the proof of the equivalence.

The proof of the second statement is similar, so we omit it. □

It is noteworthy that the assumption that no real type occurs in both $S_1$ and $S_2$ cannot be dropped. This follows directly from our discussion below Theorem 35 where we argued that a virtual substitution of a parametric root description $(f, S)$ such that $S$ contains duplicate real types is not reasonable.

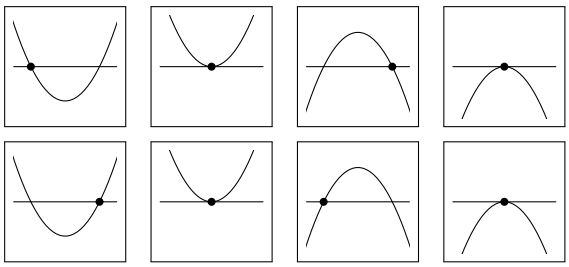


Figure 2.4: Two root specification clusters for $f = ax^2 + bx + c$: The cluster at the first line corresponds to $(f, \{(1,1),(2,1),(-1,2),(-2,1)\})$; in this case the root can be expressed as $\frac{-b-\sqrt{\Delta}}{2a}$. The cluster at the second line corresponds to $(f, \{(1,2),(2,1),(-1,1),(-2,1)\})$; in this case the root can be expressed as $\frac{-b+\sqrt{\Delta}}{2a}$.

To lift now the idea of clustering to the quadratic case we investigate all the four real types of $f = ax^2 + bx + c$, where $a$, $b$, and $c \in \mathbb{Z}[\mathbf{u}]$, when $f$ has a real root. Let $\mathbf{a} \in \mathbb{R}^m$ be such that $f\langle\mathbf{a}\rangle$ has a real root, i.e., $f\langle\mathbf{a}\rangle$ is of real 2-type 1, 2, $-1$, or $-2$. Then the marked root of $f\langle\mathbf{a}\rangle$ depicted at the first line of Figure 2.4 can be expressed in terms of the parameters $\mathbf{a}$ as

$$\frac{-b\langle\mathbf{a}\rangle - \sqrt{b\langle\mathbf{a}\rangle^2 - 4a\langle\mathbf{a}\rangle c\langle\mathbf{a}\rangle}}{2a\langle\mathbf{a}\rangle}.$$

Similarly, each marked root of $f\langle\mathbf{a}\rangle$ depicted at the second line of Figure 2.4 can be expressed as

$$\frac{-b\langle\mathbf{a}\rangle + \sqrt{b\langle\mathbf{a}\rangle^2 - 4a\langle\mathbf{a}\rangle c\langle\mathbf{a}\rangle}}{2a\langle\mathbf{a}\rangle}.$$

Using this insight and substituting square-root expressions into an atomic formula $g \varrho 0$, where $g \in \mathbb{Z}[\mathbf{u}][x]$, $\deg g < 2$, and $\varrho \in \{=, <, \leq\}$, as described in [84, Section 2], we derive formula schemes for virtual substitution with clustering. We list all the obtained schemes after simplification in Appendix A.3. Results of [84] ensure the correctness of all formula schemes derived in this way. Observe that the schemes there work exclusively with parametric root descriptions depicted in Figure 2.4, namely:

$$(f, \{(1,1),(2,1),(-1,2),(-2,1)\}) \quad \text{and} \quad (f, \{(1,2),(2,1),(-1,1),(-2,1)\}).$$

We show that these two parametric root descriptions are sufficient to realize quadratic virtual substitution with clustering on an example: Consider a quadratic atomic formula $f \geq 0$, where $f = ax^2 + bx + c$ and $a$, $b$, $c \in \mathbb{Z}[\mathbf{u}]$ contained in a formula $\varphi$ to which we apply an instantiation of `vs-scheme`. Algorithm `vs-at` returns the following candidate solutions for this atomic formula:

$$\begin{aligned}&(f,(1,1),\mathrm{WUB}),(f,(1,2),\mathrm{WLB}),\\&(f,(-1,1),\mathrm{WLB}),(f,(-1,2),\mathrm{WUB}),(f,(-2,1),\mathrm{IP}),\\&(\mathrm{red}\, f,(1,1),\mathrm{WLB}),(\mathrm{red}\, f,(-1,1),\mathrm{WUB}).\end{aligned}$$

Assume that `vs-scheme` simply selects lower bounds. Therefore, we have to substitute parametric root descriptions $(f,(1,2))$, $(f,(-1,1))$, and $(f,(-2,1))$.

Instead of computing

$$\gamma_1 \wedge \varphi[x /\!\!/ (f,(1,2))] \vee \gamma_{-1} \wedge \varphi[x /\!\!/ (f,(-1,1))] \vee \gamma_{-2} \wedge \varphi[x /\!\!/ (f,(-2,1))],$$

where $\gamma_1$, $\gamma_{-1}$, and $\gamma_{-2}$ are guards for $(f,(1,2))$, $(f,(-1,1))$, and $(f,(-2,1))$, respectively, we would like to employ clustering and rather compute formula

$$\gamma_{\{1,-1,-2\}} \wedge \varphi[x /\!\!/ (f,\{(1,2),(-1,1),(-2,1)\})],$$

where $\gamma_{\{1,-1,-2\}}$ is a guard of $(f,\{(1,2),(-1,1),(-2,1)\})$. Observe, however, that we can use our schemes from Appendix A.3, and compute

$$\gamma_{\{1,2,-1,-2\}} \wedge \varphi[x /\!\!/ (f,\{(1,2),(2,1),(-1,1),(-2,1)\})]$$

instead for the following reasons: First, it is *never* a problem to substitute too many test points; the only criterion is to guarantee that all test points generated by `vs-scheme` are substituted. Therefore, the implicit substitution of $(f,(2,1))$ is correct, but seemingly redundant. Second, the substitution of $(f,(2,1))$ is for free: The formulas in Appendix A.3 were derived to hold for *all* involved root specifications. Third, we can even use a simpler guard: Instead of using a guard $\gamma_{\{1,-1,-2\}} \longleftrightarrow a > 0 \wedge b^2 - 4ac > 0 \vee a < 0 \wedge b^2 - 4ac \geq 0$ we can use a guard $\gamma_{\{1,2,-1,-2\}} \longleftrightarrow a \neq 0 \wedge b^2 - 4ac \geq 0$.

To measure the benefit of clustering in the quadratic case we consider a polynomial $f$ of degree two, and compare the numbers of atomic formulas obtained by substituting all of the following six parametric root descriptions

$$(f,(1,1)),(f,(1,2)),(f,(2,1)),(f,(-1,1)),(f,(-1,2)),(f,(-2,1)) \tag{2.5}$$

into atomic formula $g \varrho 0$, where $g \in \mathbb{Z}[\mathbf{u}][x]$ is of degree smaller than two. Table 2.3 lists these numbers. The columns "simple" list the sum of the numbers of atomic formulas in six virtual substitution results obtained when substituting (2.5) using virtual substitution formula schemes of Appendix A.1 into $g \varrho 0$. The columns "clustering" list the sum of the numbers of atomic formulas in two virtual substitution results obtained using virtual substitution formula schemes with clustering of Appendix A.3. We point to the fact that a substitution of the six parametric root descriptions (2.5) is a completely realistic example, because all the six parametric root descriptions need to be substituted into $\varphi$ whenever $f = 0$ occurs in $\varphi$. Therefore, according to Table 2.3, the overall resulting quantifier-free equivalent should be definitely shorter when we use clustering. We will measure this on practical examples in Chapter 6.

The idea of clustering in the cubic case is indeed more challenging. Compared with the degree two, we do not use any root expressions that indicate what to cluster. Nevertheless, the quadratic case motivates us to somehow minimize the sum of the lengths of the results when using clustered virtual substitution of a set of parametric root descriptions into an atomic formula. Recall that one simple cubic virtual substitution boils down to multiple linear or quadratic virtual substitutions. Our aim is to empirically minimize this number of lower-degree virtual substitutions. Using human intelligence along with machine simplifications we propose a clustering strategy that tries to minimize the number of lower-degree virtual substitutions. Our strategy clusters root specifications according to Figure 2.5. All virtual substitution formula schemes for these clustered cubic parametric root specifications are given in Appendix A.4.

| $\varrho$ | $\deg g = 0$ | | $\deg g = 1$ | |
|---|---|---|---|---|
| | simple | clustering | simple | clustering |
| $=$ | 6 | 2 | 10 | 4 |
| $<$ | 6 | 2 | 22 | 10 |
| $\leq$ | 6 | 2 | 18 | 8 |

Table 2.3: Numbers of atomic formulas obtained by virtual substitution of six parametric root descriptions (2.5) of a quadratic polynomial $f \in \mathbb{Z}[\mathbf{u}][x]$ into an atomic formula $g \; \varrho \; 0$, where $g \in \mathbb{Z}[\mathbf{u}][x]$ is a polynomial with $\deg g < 2$.

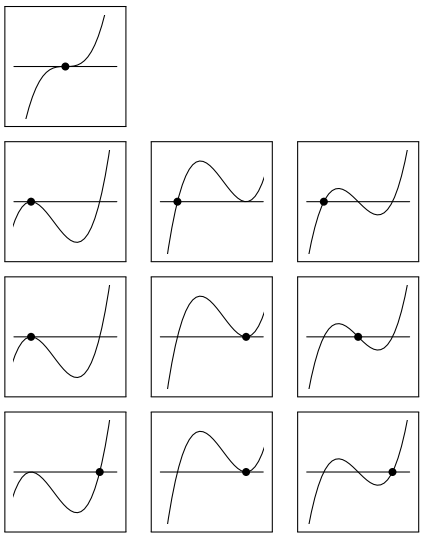


Figure 2.5: Four root specification clusters for polynomial $f = ax^3 + bx^2 + cx + d$: $(f, \{(1,1)\})$, $(f, \{(2,1),(3,1),(4,1)\})$, $(f, \{(2,1),(3,2),(4,2)\})$, and $(f, \{(2,2),(3,2),(4,3)\})$.

To prove the correctness of the schemes of Appendix A.4, one proceeds similarly as in Subsection 2.5.2. Since more than one root specification can now be present in a set of root specifications $S$, one has to ensure that a Tarski formula formalizing a condition like "the first real root of $f\langle\mathbf{a}\rangle$ lies to the right of the second real root of $g\langle\mathbf{a}\rangle$" holds whenever $f\langle\mathbf{a}\rangle$ is of one of the real types from $S$. Therefore, instead of constructing a formula for one concrete situation, we construct a formula covering more situations at once.

In Table 2.4 we summarize the numbers of lower-degree virtual substitutions of simple and clustered virtual substitution in the following setting: We count lower-degree virtual substitutions when we substitute the following eight parametric root descriptions

$$\begin{aligned}&(f,(1,1)),(f,(2,1)),(f,(2,2)),(f,(3,1)),(f,(3,2)),\\&(f,(4,1)),(f,(4,2)),(f,(4,3)),\end{aligned} \tag{2.6}$$

into an atomic formula $g \; \varrho \; 0$, whereas $f$, $g \in \mathbb{Z}[\mathbf{u}][x]$ such that $\deg f = 3$ and $\deg g < \deg f$. The columns "simple" list the sum of the numbers of lower-degree virtual substitutions needed for substitution of the eight parametric root

| $\varrho$ | $\deg g = 0$ | | $\deg g = 1$ | | $\deg g = 2$ | |
|---|---|---|---|---|---|---|
| | simple | clustering | simple | clustering | simple | clustering |
| $=$ | 0 | 0 | 14 | 8 | 28 | 18 |
| $<$ | 0 | 0 | 13 | 8 | 24 | 16 |
| $\leq$ | 0 | 0 | 13 | 8 | 25 | 17 |

Table 2.4: Numbers of lower-degree (i.e., linear and quadratic) virtual substitutions induced by virtual substitution of eight parametric root descriptions (2.6) of a cubic polynomial $f \in \mathbb{Z}[\mathbf{u}][x]$ into an atomic formula $g \ \varrho \ 0$, where $g \in \mathbb{Z}[\mathbf{u}][x]$ is a polynomial with $\deg g < 3$.

descriptions (2.6) into $g \ \varrho$ using Appendix A.2, while the columns "clustering" list the sum of the numbers of lower-degree virtual substitutions needed when substituting (2.6) using Appendix A.4. It is noteworthy that the numbers of lower-degree virtual substitutions using clustering are smaller despite the fact that some parametric root descriptions are covered more than once when using clustering. We will compare the lengths of quantifier-free equivalents obtained by simple and clustered virtual substitution on practical examples in Chapter 6.

Integrating the clustering technique developed here into our scheme is now straightforward: Instead of substituting each test point in step 7 of algorithm `vs-scheme` separately, we first partition the set $E$ of test points with respect to polynomials and infinitesimal types. Test points in a partition are then covered by the minimum number of parametric root descriptions from Figure 2.4 and Figure 2.5. Afterwards, formula schemes of Appendix A.3 and Appendix A.4 are used in algorithm `vs-at` to realize clustered virtual substitutions.

Finally, we point to the fact that all our correctness results regarding guards, virtual substitution, and infinitesimals remain valid, so that the correctness of the scheme using clustering follows from our previous results, Lemma 43, and the correctness of formula schemes of Appendix A.3 and Appendix A.4.

### 2.5.4 Towards Higher Degrees

The instantiations of `vs-scheme` described above work when the target variable $x$ has degree at most three in the input formula $\varphi$. If the degree of $x$ is greater than three, then the algorithms fail, because we did not show how to compute a guard of a higher-degree polynomial, and we did not provide an algorithm `vs-at` for substituting a parametric root description of a higher-degree polynomial. Here we discuss two topics: First, how to use the developed algorithms for degrees at most three to handle higher degrees of $x$ in some special cases. Second, how to extend the framework and provide complete instantiations beyond degree three.

We discuss algorithm `at-cs-fac`, which computes a set of candidate solutions for $f \ \varrho \ 0$ by first factorizing $f$ and applying algorithm `at-cs` afterwards to each obtained factor:

Algorithm `at-cs-fac`$(f \ \varrho \ 0, x)$.
Input: an atomic formula $f \ \varrho \ 0$, where $f \in \mathbb{Z}[\mathbf{u}][x]$ and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$.
Output: a set $\mathfrak{c}$ of candidate solutions for $f \ \varrho \ 0$.

1. If $f$ does not contain $x$, then return $\emptyset$.

2. Let $e$, $f_1^{m_1}, \ldots, f_k^{m_k}$ be a factorization of $f$ with the following properties:
   (i) $f = e \cdot f_1^{m_1} \cdots f_k^{m_k}$,
   (ii) $e \in \mathbb{Z}[\mathbf{u}]$, and
   (iii) each $f_i \in \mathbb{Z}[\mathbf{u}][x]$ has degree $d_i > 0$ and multiplicity $m_i \geq 1$.
3. If $\varrho$ is "$=$" or "$\neq$," then
   3.1. $\mathfrak{c} := \emptyset$
   3.2. For each $i \in \{1, \ldots, k\}$ do
      3.2.1. $\mathfrak{c} := \mathfrak{c} \cup \texttt{at-cs}(f_i \ \varrho \ 0, x)$
   3.3. Return $\mathfrak{c}$.
4. If $k = 1$ and the sign $s$ of $e\langle \mathbf{a} \rangle$ is constant and nonzero regardless of $\mathbf{a} \in \mathbb{R}^m$, then
   4.1. If $s = -1$, then
      4.1.1. If $\varrho$ is "$<$" ("$\leq$"), then set $\varrho$ to "$>$" ("$\geq$").
      4.1.2. If $\varrho$ is "$>$" ("$\geq$"), then set $\varrho$ to "$<$" ("$\leq$").
   4.2. If $m_1$ is even, then
      4.2.1. If $\varrho$ is "$>$," then return $\texttt{at-cs}(f_1 \neq 0, x)$.
      4.2.2. If $\varrho$ is "$\leq$," then return $\texttt{at-cs}(f_1 = 0, x)$.
      4.2.3. Return $\emptyset$.
   4.3. Return $\texttt{at-cs}(f_1 \ \varrho \ 0, x)$.
5. $\mathfrak{c} := \emptyset$
6. For each $i \in \{1, \ldots, k\}$ do
   6.1. If $m_i$ is even, then
      6.1.1. If $\varrho$ is "$<$" or "$>$," then $\mathfrak{c} := \mathfrak{c} \cup \texttt{at-cs}(f_i \neq 0, x)$.
      6.1.2. If $\varrho$ is "$\leq$" or "$\geq$," then $\mathfrak{c} := \mathfrak{c} \cup \texttt{at-cs}(f_i = 0, x)$.
   6.2. If $m_i$ is odd, then
      6.2.1. If $\varrho$ is "$<$" or "$>$," then
         $\mathfrak{c} := \mathfrak{c} \cup \texttt{at-cs}(f_i < 0, x) \cup \texttt{at-cs}(f_i > 0, x)$.
      6.2.2. If $\varrho$ is "$\leq$" or "$\geq$," then
         $\mathfrak{c} := \mathfrak{c} \cup \texttt{at-cs}(f_i \leq 0, x) \cup \texttt{at-cs}(f_i \geq 0, x)$.
7. Return $\mathfrak{c}$.

The correctness of algorithm `at-cs-fac` follows directly from Proposition 21 and Proposition 22. Observe that algorithm `at-cs-fac` fails for $f \ \varrho \ 0$ whenever $f$ has an irreducible factor of degree greater than three in $x$. To integrate algorithm `at-cs-fac` into `vs-scheme` we just call in step 3.1 `at-cs-fac` instead of `at-cs`.

One interesting aspect of algorithm `at-cs-fac` is that it takes *any* factorization of $f$ in step 2, so $f_i$ need not be irreducible. This freedom means that, e.g., a factor of degree two and a factor of degree one can be regarded as one factor of degree three. It is not clear whether something like this is helpful, or

what the best strategy looks like. Clearly, we should definitely try whether $f$ splits to factors of degrees at most three, so that we can continue instead of failing. For now we propose to factor $f$ into irreducible factors, and leave closer investigation for future work.

Another technique that can help to handle higher degrees is *degree shift*. Since the whole Chapter 4 is devoted to this technique and its generalizations, we do not discuss it here.

Now we turn to generalizations of our approach beyond degree three. Recall from Section 2.4 that for a generalization to degree $d$ we have to provide algorithms `at-cs`, `guard`, and `vs-prd-at` that are able to handle polynomials of degree $d$. Algorithm `at-cs` of Subsection 2.5.2 works for degree $d$ whenever we know all the real $d$-types. Therefore, for a successful instantiation of `vs-scheme` handling degree $d$ we need to provide:

1. Enumeration of all real $d$-types along with their respective guards, which are quantifier-free Tarski formulas in the parameters.

2. A correct substitution procedure `vs-prd-at`$\big(g \ \varrho \ 0, (f, S), x\big)$ that works for arbitrary $g$, $f \in \mathbb{Z}[\mathbf{u}][x]$, and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$ such that $\deg g < \deg f = d$, while assuming that $S$ contains exactly one root specification of $f$. For simplicity we can even assume that the leading coefficients of $g$ and $f$ are positive.

Before discussing real types and guards, we first provide evidence that algorithm `vs-prd-at` for degree $d$ can always be realized using the geometric approach of Subsection 2.5.2: Assume we are given a parametric root description $(f, (t_1, r))$ with $\deg f = d_1$ and a polynomial $g$ with $\deg g = d_2$ such that $d_2 < d_1$. The core idea is to express the necessary and sufficient conditions for $g \ \varrho \ 0$ to hold at the $r$-th real root of $f$ as a set of constraints on the relative positions of the real roots of $f\langle\mathbf{a}\rangle$ and $g\langle\mathbf{a}\rangle$ for any parameter values $\mathbf{a} \in \mathbb{R}^m$ such that $f\langle\mathbf{a}\rangle$ is of real $d_1$-type $t_1$ and $g\langle\mathbf{a}\rangle$ is of real $d_2$-type $t_2$. More specifically, given parameter values $\mathbf{a} \in \mathbb{R}^m$ such that $f\langle\mathbf{a}\rangle$ is of real $d_1$-type $t_1$ and $g$ is of real $d_2$-type $t_2$, $g \ \varrho \ 0$ holds at the $r$-th real root of $f$ if and only if each root of $g\langle\mathbf{a}\rangle$ lies in a finite union of disjoint intervals induced by the roots of $f\langle\mathbf{a}\rangle$. The actual intervals for each of the roots of $g\langle\mathbf{a}\rangle$ are indeed determined by the relative positions of $f$ and $g$. Thom's lemma [8] guarantees that we can describe these intervals in terms of sign conditions on $f$ and its derivatives. Substituting parametric root descriptions of the roots of $g$ into sign conditions on $f$ and its derivatives we then obtain necessary and sufficient conditions on $g \ \varrho \ 0$ to hold at the $r$-th real root of $f$.

We illustrate this with an example when $\deg f = 4$ and $\deg g = 2$. Assume that we are given parameter values $\mathbf{a}$ such that $f\langle\mathbf{a}\rangle$ is of real 4-type $t = (1, 0, -1, 0, 1, 0, -1, 0, 1)$ and $g\langle\mathbf{a}\rangle$ is of real 2-type 1. We derive necessary and sufficient conditions in the parameters for $(g < 0)[x \mathbin{/\!\!/} \lambda]$, $\lambda = (f, (t, 2))$, to hold under the assumptions that $f\langle\mathbf{a}\rangle$ is of real 4-type $t$ and $g\langle\mathbf{a}\rangle$ is of real 2-type 1. These assumptions ensure that $g\langle\mathbf{a}\rangle$ is negative at the second real root of $f\langle\mathbf{a}\rangle$ if and only if both of the following hold:

(i) The first real root of $g\langle\mathbf{a}\rangle$ lies to the left of the second real root of $f\langle\mathbf{a}\rangle$.

(ii) The second real root of $g\langle\mathbf{a}\rangle$ lies to the right of the second real root of $f\langle\mathbf{a}\rangle$.

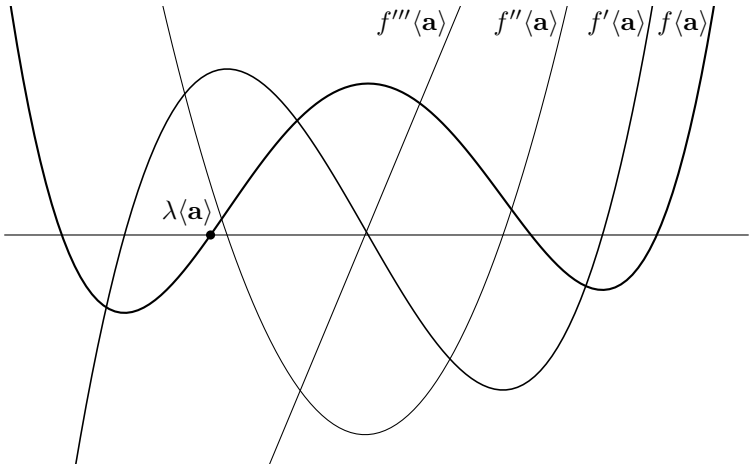



Figure 2.6: A polynomial $f\langle\mathbf{a}\rangle$ of real 4-type $t = (1, 0, -1, 0, 1, 0, -1, 0, 1)$ with its derivatives and its root $\lambda\langle\mathbf{a}\rangle = (f, (t, 2))\langle\mathbf{a}\rangle$.

Denote $(g, (1, 1))$ by $\beta_1$ and $(g, (1, 2))$ by $\beta_2$. With the help of Figure 2.6 that depicts $f\langle\mathbf{a}\rangle$ along with its derivatives it is easy to see that $\beta_1\langle\mathbf{a}\rangle$ lies to the left of $\lambda\langle\mathbf{a}\rangle$ if and only if

$$\Big(f''' < 0 \wedge \big(f < 0 \vee f' < 0\big)\Big)[x \mathbin{/\!\!/} \beta_1]$$

is satisfied by $\mathbf{a}$. Similarly, $\beta_2\langle\mathbf{a}\rangle$ lies to the right of $\lambda\langle\mathbf{a}\rangle$ if and only if

$$\Big(f''' \geq 0 \vee \big(f > 0 \wedge f''' < 0 \wedge (f' > 0 \vee f'' < 0)\big)\Big)[x \mathbin{/\!\!/} \beta_2]$$

is satisfied by $\mathbf{a}$. To sum up, we obtained a virtual substitution formula scheme that works for $\mathbf{a}$ under the assumption that $f$ is of real 4-type $t$ and $g$ is of real 2-type 1.

Observe that the approach we just discussed is a kind of bootstrapping: If one can realize virtual substitutions of parametric root descriptions of polynomials with degrees $1, \ldots, d-1$, then one can use it to realize virtual substitution for parametric root descriptions of degree $d$. We think that it should be possible to automatize this bootstrapping process to derive conditions on $f$ and its derivatives to simplify the tedious derivation of virtual substitution formula schemes. Another topic related to this is clustering. We believe that it could be also automatized to some extent, especially for some "similar" real types of $f$. At the end of the day, it should be possible to formulate the question about which parametric root specifications to cluster as an optimization problem with the target of minimizing the number of lower-degree virtual substitutions. We leave all these questions to further work.

The last thing to discuss in this context is the enumeration of all real $d$-types and the derivation of a guard for a real $d$-type. In Section 2.1 we precisely counted the number of real $d$-types by counting the number of solutions of a certain integer constraint system. We think that a clever adjustment of this technique could lead to an algorithm enumerating all the real $d$-types explicitly as sign lists. The actual work then would be to give a guard for every single real

$d$-type. Using tricks like linear transformation as in [83, Section 2] and using automated tools like Qepcad [13] this should be feasible up to degree five without much effort. As an example, we computed guards of all seven potential real 4-types of $f = x^4+ax^2+bx+c$ by running Qepcad on formulas from Proposition 8 without using preprocessing or applying some tricks or transformations. Despite this the time of the whole computation was not higher than three minutes. For higher degrees, in contrast, this naive approach will definitely not work and clever mathematical tricks will be probably needed.

To conclude this subsection we can say that the framework developed in this chapter can be instantiated to obtain a complete stand-alone quantifier elimination procedure handling degrees of the quantified variable beyond degree three. We believe that it is even possible to automatize and optimize this process. We regard the derivation of guards as the main bottleneck of any potential automated approach. Finally, other incomplete heuristics like factorization discussed here are certainly of practical interest as well.

## 2.6 Comparison with Existing Approaches

In this section we summarize the similarities and differences of our framework with existing real quantifier elimination techniques. To begin with, we look at our framework in the context of other virtual substitution-based methods.

The linear virtual substitution method [51] of Weispfenning was the first method based on the idea of substituting formal solutions of polynomials involved in an input formula $\varphi$ into the formula to eliminate an existential quantifier. Extending this idea by using root expressions as formal solutions of a quadratic polynomial led to quadratic virtual substitution [84]. The cubic virtual substitution [83], in contrast, uses real types and root indices to represent real roots of a cubic polynomial.

The main advantages of our framework compared to any of the approaches mentioned above are:

1. Independence of degrees and extensibility: Our framework makes no principal difference between a test point coming from a linear or a higher-degree atomic formula. No quotients or root expressions are part of it, so the well-known Abel-Ruffini theorem represents no principal boundary for extensibility of the framework.

2. Reducing the number of test points: All mentioned approaches treat all test points of the involved polynomials equally, and do not care much about the atomic formula generating a particular test point. The bound selection is therefore limited compared to our approach. The concept of candidate solutions sets developed in Section 2.2 enables us to represent all the parametric root descriptions needed. Consequently, we can distinguish easily between lower bounds and upper bounds, and discard redundant test points originating from atomic formulas with relation symbols $\leq$, $\geq$, $<$, and $>$.

3. Stronger guards: Compared with the linear virtual substitution we obtain stronger guards as mentioned in [25, Section 3.6]. This happens automatically thank to the candidate solutions concept, which is an intermediate step between polynomials and test points.

4. Clustering: Compared with the cubic virtual substitution we substitute more parametric root descriptions at once to obtain shorter quantifier-free equivalents.

Another idea of Weispfenning to use Thom codes to represent roots of a polynomial, and use these for virtual substitution were made explicit in our publication [46]. We discuss its relation to our framework here below in a separate subsection.

One of our main motivations for the development of a framework for virtual substitution was the lack of theoretical concepts and data structures that would provide a basis for an easily extensible and modular implementation. In this chapter we filled this gap. An implementation based on the framework developed here will be the subject of Chapter 6. There we will present the first implementation of a virtual substitution-based quantifier elimination algorithm for degree three and discuss its applications. It will turn out that the implementation based on our framework is easily extensible beyond degree three; a property inherited from the framework developed in this chapter.

We again emphasize that our goal here was not to develop an asymptotically faster approach that would beat the existing approaches. We focus on rather practical aspects and heuristics related to quantifier elimination. Our aim is not to compete with other methods achieving better worst-case complexity.

Now we briefly discuss the relation of our framework with quantifier elimination by CAD. As we argued above, our framework subsumes and improves upon all existing quantifier elimination methods based on virtual substitution. These methods were shown to perform much better than CAD especially on problems involving low-degree variables and a higher number of parameters. Therefore, these observations hold for our framework here as well. Nevertheless, for problems involving higher-degree variables, CAD remains the only practical choice. We think that the most successful general strategy is: First use virtual substitution, i.e., our framework, to eliminate as much low-degree variables as possible. If this is not successful, and some quantifiers remain because of high degrees of quantified variables, then apply CAD.

### 2.6.1 A Few Remarks on a Thom Code-Based Framework

Starting from a suggestion by Weispfenning [84, Section 6], we developed in our recent publication [46] a framework for virtual substitution based on Thom codes. A test point in that framework is a root of $f \in \mathbb{Z}[\mathbf{u}][x]$ represented as a sign sequence $\overline{s}$ of length $\deg f$. A guard of a test point $(f, \overline{s})$ and virtual substitution of $(f, \overline{s})$ are defined as follows: A guard is a quantifier-free formula in the parameters $\mathbf{u}$ that gives necessary and sufficient conditions for the existence of a root of $f$ with signs of the derivatives of $f$ being $\overline{s}$. The virtual substitution of a test point $(f, \overline{s})$ into an atomic formula yields necessary and sufficient conditions in the parameters $\mathbf{u}$ for the atomic formula to hold at the root of $f$ described by the sign sequence $\overline{s}$. The framework uses an external real quantifier elimination algorithm $\mathcal{A}$ to compute guards and virtual substitution of test points. Practical experiments suggest that the framework cannot compete with the existing quadratic and cubic virtual substitution approaches.

Here we would like to point to several differences between the framework of [46] and our framework developed in this chapter:

1. According to [46, Lemma 3], the virtual substitution of $(f, \overline{s})$ into an atomic formula yields a formula that implies a guard of $(f, \overline{s})$. This leads in practice to many redundant formulas in virtual substitution results, which are hard to detect and simplify by standard simplification techniques. It turns out that a guard of $(f, \overline{s})$ can be "masked" in a virtual result, so it is not easy to detect such guard.

   Our framework here, in contrast, uses a guard of a parametric root description exactly once and writes it conjunctively in front of the whole formula obtained by virtual substitution.

2. The framework of [46] makes no link between virtual substitution for degree $d$ and degree $d+1$. Virtual substitution formula schemes for degree $d+1$ need to be derived from scratch without taking advantage of existing virtual substitution. This makes a practical extension of the framework to higher degrees inconvenient.

   We saw that we take advantage of lower-degree virtual substitutions in all of our instantiations of `vs-scheme`. An instantiation of `vs-scheme` for degree four or higher will definitely do so as well.

3. The test points of the form $(f, \overline{s})$ reveal no geometric information about $f$, so it is not easy to interpret what is actually being substituted. This makes the development of clustering or other test point reducing techniques harder.

4. The number $3^d - 1$ of test points for a $d$-degree atomic formula seems to be too high. For example, our framework would generate for a $d$-degree atomic formula, $d = 1, 2, 3, 4$ at most 2, 8, 24, 60 test points in total, respectively. These numbers suggest that our framework here naturally merges more test points of the framework of [46] even without using clustering.

For these reasons the framework of [46] is not the right choice from the practical point of view. The mentioned deficiencies and limitations of that framework actually served as a starting point for the framework developed in this chapter. Of course, further theoretical and practical investigations of the framework of [46] and our framework here could lead to improvements on both sides. We leave this for future work.

## 2.7 Conclusions

In this chapter we developed a framework for virtual substitution. We first introduced real types, parametric root descriptions, and candidate solutions. All these concepts are used to represent, analyze, and generate test points for an input Tarski formula $\varphi$. A test point is a representation of a root of a polynomial contained in $\varphi$. We formally defined virtual substitution of a test point and proven its existence and semantic correctness. Using all these concepts we then presented an abstract approach `vs-scheme` and proven its correctness. The scheme was then instantiated by providing three precisely specified algorithms. In this way we obtained complete and self-contained quantifier elimination algorithms for degree at most three. The correctness of the algorithms is automatically guaranteed by the correctness of `vs-scheme`. All of these algorithms were

shown to subsume and improve on all existing virtual substitution approaches. Extensibility of the scheme beyond degree three and its properties were discussed afterwards, and concrete ideas on extending it to degree four were given. Finally, we compared our framework with other existing approaches.

# Chapter 3

# Structural Virtual Substitution

In this chapter we explore the potential of exploiting the Boolean structure of a quantifier-free Tarski formula $\varphi$ during quantifier elimination of $\exists x$ from $\exists x(\varphi)$. The idea to take the Boolean structure into account during the process of elimination of a single quantifier was first mentioned by Dolzmann in [25]. He introduced the concepts of "structural elimination sets" and "condensing." Similar concepts in the context of Presburger arithmetic were studied in [48]. Building on this previous work, we propose new improvements like our Marking technique and novel bound selection strategies. We give concrete algorithms that we integrate into our framework for virtual substitution presented in Chapter 2.

The starting point for our discussion here is the quantifier elimination algorithm scheme presented and proven correct in Chapter 2, which can be described as a sequence of the following three phases:

**Phase 1** Extract from $\varphi$ the set $S$ of all atomic formulas containing $x$. Analyze each atomic formula in $S$ separately: Factorize the left hand side, and compute a set of candidate solutions for the atomic formula by appropriately using the obtained factors. Collect all candidate solutions obtained this way into one set $\mathfrak{c}$, which constitutes a set of candidate solutions for $\varphi$ and $x$.

**Phase 2** From $\mathfrak{c}$ select either the candidate solutions representing the upper bounds or the candidate solutions representing the lower bounds. Convert the selected candidate solutions into a set of test points $E$ adjusting by $\pm\varepsilon$ where needed and finally adding $\mp\infty$. The set $E$ is an elimination set for $\varphi$ and $x$.

**Phase 3** Substitute each test point from $E$—preferably using clustering—into $\varphi$ by means of virtual substitution, and construct a disjunction of the results. Since $E$ is an elimination set for $\varphi$ and $x$, the obtained formula is a quantifier-free equivalent of $\exists x(\varphi)$.

In this chapter we are going to make every single of the above steps sensitive to the Boolean structure of $\varphi$. We will integrate a careful analysis of the Boolean

structure of the input formula $\varphi$ into our quantifier elimination algorithm scheme of Section 2.4.

Instead of throwing all atomic formulas of $\varphi$ into one pot in Phase 1, we first decompose $\varphi$ into a set of so-called prime constituents. A prime constituent represents either an atomic or a more complex subformula of $\varphi$ along with a set of candidate solutions for that particular subformula. The idea is to group atomic formulas based on the Boolean structure of $\varphi$, so that some atomic formulas are not taken into account during the computation of a candidate solution set for $\varphi$ at all. We will identify equations and negated equations at certain particularly favorable positions within $\varphi$ in order to identify non-atomic prime constituents, which will produce significantly fewer candidate solutions than the atomic formulas contained in there.

We abandon the global bound selection paradigm in Phase 2. We instead propose a more liberal bound selection strategy using 0-1 Integer Linear Programming that is not restricted to the "either take all upper bounds or all lower bounds" approach. It will be possible that our strategy picks lower bounds from some prime constituent and upper bounds from another. To achieve this, the Boolean structure of $\varphi$ will be taken into account to detect dependent subformulas w.r.t. bound selection. To analyze the dependency, we will introduce the notion of conjunctive associativity.

Substituting a test point into $\varphi$ during Phase 3 we will take advantage of the Boolean structure of $\varphi$ again. For this we assign to each test point its structural origin, i.e., the prime constituent of $\varphi$ that generated that test point during the test points generation process. This structural origin is then used during the substitution to trim $\varphi$—it turns out that a test point needs not to be substituted into the whole $\varphi$ in some special cases; some atomic formulas of $\varphi$ can be simply replaced with "false" instead of applying a costly virtual substitution to them. The above-mentioned notion of conjunctive associativity will be playing a central role here as well. Furthermore, to argue the correctness of our approach we will be looking at a particular DNF of $\varphi$.

Observe that even when we focus here exclusively on the elimination of a single variable and that our techniques presented here benefit more from a richer Boolean structure of an input formula, the results of this chapter are relevant in a broader context: As we have seen in Chapter 2, virtual substitution introduces richer Boolean structure while eliminating a quantifier. Therefore, successive quantifier elimination steps can exploit the Boolean structure of an intermediate result obtained by virtual substitution. In this way the techniques of this chapter become applicable even when one starts with a seemingly simple formula having a flat Boolean structure; whenever the target is to eliminate more than one quantifier it is possible that intermediate results will exhibit properties that can be exploited by techniques of this chapter. In fact, many applications of quantifier elimination and decision in practice ask for elimination of significantly more than one variable.

## 3.1 Prime Constituent Decompositions

Let $\varphi$ be a quantifier-free Tarski formula, which is an $\wedge$-$\vee$-combination of atomic formulas. In the following we represent the Boolean structure of $\varphi$ by *the structural tree for* $\varphi$. It is a rooted tree whose leaves correspond to the atomic

formulas occurring in $\varphi$. Inner nodes of the tree are either $\wedge$-nodes or $\vee$-nodes, whereas the children of a node represent the operands of the respective Boolean operator in the same order as they appear in $\varphi$. The root of the tree is the top-level Boolean operator of $\varphi$. If $\varphi$ is an atomic formula, then the root is defined to be this atomic formula. A subtree of the structural tree for $\varphi$ represents a subformula of $\varphi$. Note "true" and "false" are atomic formulas as well.

A *position* $\pi = (p_1, \ldots, p_k)$ is a finite, possibly empty, sequence of nonzero natural numbers $p_i$ identifying a particular node in a structural tree. The empty position is denoted by "()." If $\pi$ represents a leaf of the tree, then we call it *leaf* or *atomic* position. Otherwise we call $\pi$ an *inner* position. We denote the concatenation operator by "|," so for a position $\pi = (p_1, \ldots, p_k)$ and nonzero natural $p$ we have $\pi|p = (p_1, \ldots, p_k, p)$, and $()|p = (p)$. The set of all valid positions in $\varphi$ is denoted by $\texttt{Pos}(\varphi)$.

By $\pi(\varphi)$ we denote the subtree of the structural tree for $\varphi$ rooted at position $\pi$. We overload the notation and denote by $\pi(\varphi)$ also the subformula of $\varphi$ represented by the subtree $\pi(\varphi)$. It will be always clear from the context whether $\pi(\varphi)$ stands for the subtree or for the subformula of $\varphi$ at position $\pi$.

Let $\pi_1 = (p_1, \ldots, p_k)$ and $\pi_2 = (r_1, \ldots, r_l)$ be two positions. The *lowest common ancestor* of $\pi_1$ and $\pi_2$, denoted by $\mathrm{lca}(\pi_1, \pi_2)$, is the position $(p_1, \ldots, p_m)$, where $m \in \{1, \ldots, \min(k, l)\}$ is maximal with the property $p_i = r_i$ for all $i \in \{1, \ldots, m\}$. If $p_1 \neq r_1$, then $\mathrm{lca}(\pi_1, \pi_2)$ is $()$. If $\mathrm{lca}(\pi_1, \pi_2) = \pi_1$ and $\pi_1 \neq \pi_2$, then we say that $\pi_1$ is a *parent position* of $\pi_2$, or equivalently $\pi_2$ is a *subposition* of $\pi_1$. We denote that $\pi_2$ is a subposition of $\pi_1$ by $\pi_2 \sqsubset \pi_1$. Note that $\pi_2 \sqsubset \pi_1$ directly implies that $\pi_2(\varphi)$ is a subformula of $\pi_1(\varphi)$. If $\pi_1 = \pi_2$ or $\pi_2 \sqsubset \pi_1$, then we write $\pi_2 \sqsubseteq \pi_1$. If not $\pi_2 \sqsubseteq \pi_1$, then we write $\pi_2 \not\sqsubseteq \pi_1$. If $\pi_1 \not\sqsubseteq \pi_2$ and $\pi_2 \not\sqsubseteq \pi_1$, then we say that positions $\pi_1$ and $\pi_2$ are *independent*. Observe that $\mathrm{lca}(\pi_1, \pi_2)$ of two independent positions is always an inner node of a structural tree.

**Example 44.** To make our definitions clear we illustrate them on an example. Consider a quantifier-free Tarski formula $\varphi$:

$$\big((x - a = 0 \wedge x - c \geq 0) \vee x + 1 < 0\big) \wedge \big(x - 1 > 0 \vee (x - 1 = 0 \wedge x - b \leq 0)\big).$$

In Figure 3.1 we can see the structural tree for $\varphi$. The set $\texttt{Pos}(\varphi)$ is

$$\big\{\, (), (1), (2), (1,1), (1,2), (2,1), (2,2), (2,2), (1,1,1), (1,1,2), (2,2,1), (2,2,2) \,\big\}.$$

The subformula at position $(2,1)$ is $x - 1 > 0$, formally $(2,1)(\varphi)$ yields $x - 1 > 0$. The positions $(1,1,1)$ and $()$ are not independent, more specifically $()$ is a parent position of the atomic position $(1,1,1)$, formally $(1,1,1) \sqsubset ()$. All parent positions of $(1,1,1)$ are $(1,1)$, $(1)$, and $()$. Positions $(1,1,2)$ and $(1,2)$ are independent because $(1,1,2) \not\sqsubseteq (1,2)$ and $(1,2) \not\sqsubseteq (1,1,2)$. The lowest common ancestor of $(1,1,2)$ and $(1,2)$ is $(1)$, formally $\mathrm{lca}\big((1,1,2),(1,2)\big) = (1)$. The inner node $(1)$ is an $\vee$-node and the formula $(1)(\pi)$ is $(x - a = 0 \wedge x - c \geq 0) \vee x + 1 < 0$. Children positions of $(1)$ are $(1)|1 = (1,1)$ and $(1)|2 = (1,2)$. They represent subformulas $x - a = 0 \wedge x - c \geq 0$ and $x + 1 < 0$ respectively. $\diamond$

### 3.1.1 Prime Constituents

To generate a set of test points for an input formula $\varphi$, the framework of Chapter 2 considers a set of candidate solutions for every atomic formula occurring in

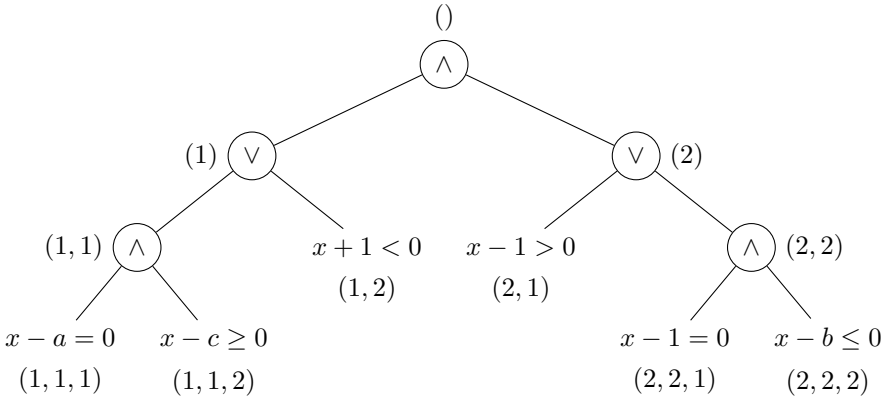



Figure 3.1: The structural tree for formula $\varphi$ from Example 44. Each node is labeled with its position in the tree.

$\varphi$. Here we introduce the notion of a prime constituent, which enables us to also consider more complex subformulas of $\varphi$, and obtain a set of candidate solutions for them directly. In this way we will be able to discard candidate solutions of some atomic formulas that would have been considered by `vs-scheme`, and would have yielded test points afterwards. Equations and negated equations will play a prominent role here. As usual we work in the context where we deal with the elimination of $\exists x$ from $\exists x(\varphi(\mathbf{u}, x))$. Recall that $\varphi$ is an $\wedge$-$\vee$-combination of atomic formulas in the Tarski language $\mathfrak{L}$ and $\mathbf{u} = (u_0, \ldots, u_{m-1})$ are parameters. We begin with two definitions.

Let $G(\mathbf{u}, x)$ be a Tarski formula such that

(i) there exists a candidate solution set $\mathfrak{c} = \big\{(f_1, S_1, \mathrm{IP}), \ldots, (f_k, S_k, \mathrm{IP})\big\}$, consisting exclusively of "IP" candidate solutions, for $G(\mathbf{u}, x)$, and

(ii) for any parameter values $\mathbf{a} \in \mathbb{R}^m$ the satisfying set $\Phi(G, \mathbf{a})$ is finite.

Then we say that $G(\mathbf{u}, x)$ is a *Gauss formula*. The parametric root descriptions $(f_i, S_i)$ are called *Gauss solutions*. Note that a Gauss formula can be unsatisfiable.

Let $C(\mathbf{u}, x)$ be a Tarski formula such that

(i) there exists a candidate solution set $\mathfrak{c} = \big\{(f_1, S_1, \mathrm{EP}), \ldots, (f_k, S_k, \mathrm{EP})\big\}$, consisting exclusively of "EP" candidate solutions, for $C(\mathbf{u}, x)$, and

(ii) for any parameter values $\mathbf{a} \in \mathbb{R}^m$ the satisfying set $\Phi(C, \mathbf{a})$ is co-finite, i.e., $\mathbb{R} \setminus \Phi(C, \mathbf{a})$ is finite.

Then we say that $C(\mathbf{u}, x)$ is a *co-Gauss formula*. The parametric root descriptions $(f_i, S_i)$ are called *co-Gauss exception points*. Note that a co-Gauss formula can be valid. Furthermore, observe that no Tarski formula can be Gauss and co-Gauss at the same time.

Natural candidates for Gauss and co-Gauss formulas are indeed equations and negated equations, respectively. Such atomic formulas have candidate solution sets consisting exclusively of "IP" and "EP" candidate solutions, respectively. However, if the left hand side of such atomic formula vanishes for some parameter values, then point (ii) of either of the two definitions is not satisfied:

**Example 45.** Atomic formula $f = 0$, where $f = u_1x + u_0 \in \mathbb{Z}[u_0, u_1][x]$ is *not* a Gauss formula even though $\big\{(f, (1,1), \mathrm{IP}), (f, (-1,1), \mathrm{IP})\big\}$ is a set of candidate solutions for $f = 0$. The reason is that for parameter values $(0,0)$ the polynomial $f$ vanishes, so $\Phi\big(f = 0, (0,0)\big) = \mathbb{R}$, which is obviously an infinite set.

Consider now an atomic formula $f \neq 0$, where $f = ax^2 + (a+1)x + b \in \mathbb{Z}[\mathbf{u}][x]$ and $a$, $b$ are arbitrary polynomials from $\mathbb{Z}[\mathbf{u}]$. Algorithm `at-cs` of Subsection 2.5.2 obviously computes a set of candidate solutions for $f \neq 0$ consisting exclusively of "EP" candidate solutions. Since $a$ and $a+1$ cannot vanish simultaneously, the satisfying set $\Phi(f \neq 0, \mathbf{a})$ is infinite for any parameter values $\mathbf{a}$. Therefore, we conclude that this formula is co-Gauss.

As the third example we consider a non-atomic formula $f = 0 \wedge \psi$, where $f = x^3 + bx - 7$, for some $b \in \mathbb{Z}[\mathbf{u}]$, and $\psi$ is an arbitrary quantifier-free Tarski formula. Obviously, there exists a set of "IP" candidate solutions for $f = 0$, and since $f$ is identically zero for no parameter values, the formula $f = 0$ is a Gauss formula. Observe now that for *any* parameter values $\mathbf{a}$ the formula $f = 0 \wedge \psi$ can possibly hold only at the roots of $f\langle\mathbf{a}\rangle$. Consequently, the whole formula $f = 0 \wedge \psi$ is a Gauss formula with Gauss solutions obtained for the atomic formula $f = 0$. ◇

Example 45 illustrates a phenomenon we would like to take advantage of: If a Gauss formula occurs in a conjunction, then the whole conjunction is a Gauss formula. Similarly, if a co-Gauss formula occurs in a disjunction, then the whole disjunction is a co-Gauss formula. Before presenting an algorithm for finding Gauss and co-Gauss subformulas in the input formula $\varphi$, we introduce the central notion of this chapter:

A *prime constituent* (PC) $\mu$ of formula $\varphi(\mathbf{u}, x)$ is a tuple $(\pi, \Upsilon, \mathfrak{c}, \mathcal{F}, B)$ with the following properties:

(i) The position of $\mu$ in $\varphi$ is $\pi$. The subformula $\pi(\varphi)$ contains $x$.

(ii) The *type* $\Upsilon$ of $\mu$ is exactly one of the following:

  (a) a Gauss formula ($\Upsilon = \mathrm{GAUSS}$),

  (b) a co-Gauss formula ($\Upsilon = \mathrm{COGAUSS}$),

  (c) an atomic formula ($\Upsilon = \mathrm{AT}$).

(iii) The set $\mathfrak{c}$ is a set of candidate solutions for the formula $\pi(\varphi)$. If we were unable to determine a set of candidate solutions for $\pi(\varphi)$, then $\mathfrak{c}$ is FAILED. We will explain the exact meaning of a failure below.

(iv) A *false replacement set* $\mathcal{F}$ is a finite set of pairwise independent positions. The set $\mathcal{F}$ is also independent with $\pi$, i.e., $\pi \not\sqsubseteq \pi'$ and $\pi' \not\sqsubseteq \pi$ for every $\pi' \in \mathcal{F}$.

(v) A *bound choice* $B$ is a subset of $\{l, u\}$.

A *prime constituent decomposition of* $\varphi$ (PC decomposition) is a set $P$ of prime constituents such that:

(i) The positions of the prime constituents in $P$ are pairwise independent.

(ii) If $\alpha$ is a position of an atomic formula in $\varphi$ containing $x$, then there exists exactly one PC in $P$ with position $\pi$ such that $\alpha \sqsubseteq \pi$.

Recall the formula $\varphi$ from Figure 3.1:

$$\big((x - a = 0 \wedge x - c \geq 0) \vee x + 1 < 0\big) \wedge \big(x - 1 > 0 \vee (x - 1 = 0 \wedge x - b \leq 0)\big).$$

One natural PC decomposition of $\varphi$ is obtained by taking all the atomic formulas in $\varphi$. This leads to a set of atomic prime constituents with positions

$$\big\{(1,1,1),(1,1,2),(1,2),(2,1),(2,2,1),(2,2,2)\big\}.$$

Another PC decomposition of $\varphi$ is obtained by discovering the two Gauss subformulas in $\varphi$ with positions $(1,1)$ and $(2,2)$. This yields a PC decomposition with positions and types as follows:

$$\big\{((1,1),\text{GAUSS}),((1,2),\text{AT}),((2,1),\text{AT}),((2,2),\text{GAUSS})\big\}.$$

Here we only mentioned the position and the type of prime constituents. The role of the entities $\mathfrak{c}$, $\mathcal{F}$, $B$, how to compute and use them will become clear later.

Before we present and prove correct our PC decomposition algorithm, let us discuss the FAILED case. In the following we will use our infrastructure developed in Chapter 2. In particular, we will use `at-cs-fac` to compute a set of candidate solutions for prime constituents of $\varphi$. The FAILED case can occur only as a result of `at-cs-fac`; whenever FAILED is returned by some algorithm in this chapter, this can be traced back to a `at-cs-fac` call that returned FAILED.

In the following we therefore slightly change the specification of `at-cs-fac`: Whenever `at-cs-fac` returns a set of candidate solutions $\mathfrak{c}$ we guarantee that we are able to substitute *any* parametric root descriptions occurring in $\mathfrak{c}$ by means of virtual substitution given in Chapter 2. This behavior was there ensured simply by assuming that the degree of $x$ in $\varphi$ is not greater than three. Here we do not bound the degree of $x$ in $\varphi$ explicitly; we allow all our algorithms to return FAILED instead.

The real reason for failure is therefore an occurrence of a high-degree irreducible factor of a left hand side of an atomic formula in $\varphi$. Since we do not use any external quantifier elimination methods, "a high-degree factor" currently means an irreducible factor of degree greater than three. As discussed in Chapter 2, this bound is not inherent and can be extended to arbitrary but bounded degrees by providing suitable algorithms for computing guards and virtual substitutions.

Finally, we mention that an occurrence of a high-degree irreducible factor does not necessarily mean that the algorithms here will fail. We will see cases for which our structural approach will be able to handle them even without higher-degree virtual substitution. Before showing such example, we continue by presenting our PC decomposition algorithm.

### 3.1.2 Computing a PC Decomposition

Algorithm `PC-decomposition` presented here computes a PC decomposition of the input formula $\varphi$. The algorithm first computes positions of Gauss, co-Gauss, and *all* atomic subformulas of $\varphi$ containing $x$ (lines 1–3). For this it uses algorithms `gaussposl`, `cogaussposl`, and `atposl` given below. Each of these

three algorithms returns a set of pairwise independent positions. Put together, however, these positions will be dependent whenever at least one Gauss or co-Gauss subformula was discovered. We therefore filter these dependent positions and decide which of them to include in the final PC decomposition.

This is done at lines 4–6. The strategy there is that "bigger and special wins," i.e., whenever a position of one type is a parent of or equal to another position the latter position is deleted. Furthermore, we prefer Gauss and co-Gauss formulas to atomic formulas. In steps 4–5 we intentionally use "$\sqsubset$" instead of "$\sqsubseteq$," because a formula cannot be simultaneously Gauss and co-Gauss by definition. The reason why we always pick the "bigger" position (a position representing a bigger subformula) will become clear at the end of this subsection where we give an example run of `PC-decomposition`.

Lines 8–10 of the algorithm construct prime constituents from the computed positions, while storing all the necessary information into each prime constituent. Here we point to the fact that an implicit ordering of Gauss positions is used. Preceding Gauss positions of a prime constituent constitute the false replacement set of the prime constituent. Here note that a Gauss prime constituent contains positions of the preceding Gauss positions, and co-Gauss and atomic prime constituents contain positions of all Gauss prime constituents. All this information will be used later by our quantifier elimination algorithm scheme in Section 3.3.

Algorithm `PC-decomposition`$(\varphi, x)$.
Input: an $\wedge$-$\vee$-combination of Tarski atomic formula $\varphi$, a variable $x$.
Output: a PC decomposition $P$ of $\varphi$, or FAILED.

1. `gpl` := `gaussposl`$(\varphi, x, ())$
2. `cgpl` := `cogaussposl`$(\varphi, x, ())$
3. `atpl` := `atposl`$(\varphi, x, ())$
4. For each Gauss position $\pi \in$ `gpl` do
   4.1. If $\pi \sqsubset \varpi$ for some $\varpi \in$ `cgpl`, then delete $\pi$ from `gpl`.
5. For each co-Gauss position $\varpi \in$ `cgpl` do
   5.1. If $\varpi \sqsubset \pi$ for some $\pi \in$ `gpl`, then delete $\varpi$ from `cgpl`.
6. For each atomic position $\alpha \in$ `atpl` do
   6.1. If $\alpha \sqsubseteq \beta$ for some $\beta \in$ `gpl` $\cup$ `cgpl`, then delete $\alpha$ from `atpl`.
7. If there exists $\pi \in$ `gpl` $\cup$ `cgpl` $\cup$ `atpl` such that $\mathfrak{c}(\pi)$ is FAILED, then
   7.1. Return FAILED.
8. Let `gpl` be $\{\pi_1, \ldots, \pi_n\}$. For $i := 1$ to $n$ do
   8.1. Let $\mathfrak{c}(\pi_i)$ be the set of candidate solutions for the formula $\pi_i(\varphi)$ computed by `gaussposl`.
   8.2. Add $(\pi_i, \text{GAUSS}, \mathfrak{c}(\pi_i), \{\pi_1, \ldots, \pi_{i-1}\}, \emptyset)$ to $P$.
9. Let `cgpl` be $\{\varpi_1, \ldots, \varpi_m\}$. For $i := 1$ to $m$ do

9.1. Let $\mathfrak{c}(\varpi_i)$ be the set of candidate solutions for the formula $\varpi_i(\varphi)$ computed by `cogaussposl`.

9.2. Add $(\varpi_i, \text{COGAUSS}, \mathfrak{c}(\varpi_i), \{\pi_1, \ldots, \pi_n\}, \emptyset)$ to $P$.

10. Let `atpl` be $\{\alpha_1, \ldots, \alpha_l\}$. For $i := 1$ to $l$ do

10.1. Let $\mathfrak{c}(\alpha_i)$ be the set of candidate solutions computed for the formula $\alpha_i(\varphi)$ by `atposl`.

10.2. Add $(\alpha_i, \text{AT}, \mathfrak{c}(\alpha_i), \{\pi_1, \ldots, \pi_n\}, \emptyset)$ to $P$.

11. Return $P$.

The aim of algorithms `gaussposl` and `cogaussposl` is to discover Gauss and co-Gauss subformulas of the input formula $\varphi$. We do *not* guarantee that all Gauss or all co-Gauss subformulas are found. If we wanted to guarantee this, then we would have to decide for each subformula $\pi(\varphi)$ of $\varphi$ whether the solution set $\Phi(\pi(\varphi), \mathbf{a})$ is finite/infinite for any parameter values $\mathbf{a} \in \mathbb{R}^m$, which is indeed too costly.

Here we focus on the Boolean structure of $\varphi$ instead. The core idea of algorithm `gaussposl` is as follows: We recursively traverse $\varphi$ and whenever we find a Gauss formula occurring in a conjunction we deduce that the whole conjunction is a Gauss formula as well. If there are more Gauss conjuncts, we are free to pick one of them with the best candidate solution set. For a disjunction, in contrast, all disjuncts have to be Gauss.

Algorithm `gaussposl`$(\varphi, x, \pi)$.
Input: an $\wedge$-$\vee$-combination of Tarski atomic formulas $\varphi$, a variable $x$, a position $\pi$ in $\varphi$.
Output: a set $\{\pi_1, \ldots, \pi_n\}$ of pairwise independent positions such that each $\pi_j(\varphi)$ is a Gauss formula; for each $j \in \{1, \ldots, n\}$ we also obtain a set of candidate solutions $\mathfrak{c}(\pi_j)$ for the formula $\pi_j(\varphi)$ or FAILED, which means that we were unable to compute a set of candidate solutions for the Gauss subformula $\pi_j(\varphi)$.

1. If $\pi(\varphi)$ is an atomic formula $f \varrho 0$, then

1.1. If $\varrho$ is "=," $x$ occurs in $f \in \mathbb{Z}[\mathbf{u}][x]$, and the coefficients of $f$ cannot all vanish simultaneously, then

1.1.1. $\mathfrak{c}(\pi) :=$ `at-cs-fac`$(f = 0, x)$

1.1.2. Return $\{\pi\}$.

1.2. Return $\emptyset$.

2. For each child position $\pi|1$, ..., $\pi|n$ of $\pi$ do

2.1. $G_i :=$ `gaussposl`$(\varphi, x, \pi|i)$

3. Denote by $G$ the set of all Gauss child positions of $\pi$, i.e., $\pi|i \in G$ if and only if $G_i = \{\pi|i\}$.

4. If the top-level operator of $\pi(\varphi)$ is "$\wedge$" and $|G| > 0$, then

4.1. Select one $\pi|i \in G$ with "the best" $\mathfrak{c}(\pi|i)$.

4.2. $\mathfrak{c}(\pi) := \mathfrak{c}(\pi|i)$

4.3. Return $\{\pi\}$.

5. If the top-level operator of $\pi(\varphi)$ is "$\vee$" and $|G| = n$, then

   5.1. $\mathfrak{c}(\pi) := \bigcup_{\pi|i \in G} \mathfrak{c}(\pi|i)$

   5.2. Return $\{\pi\}$.

6. Return $\bigcup_{i=1}^{n} G_i$.

Algorithm `cogaussposl` is a kind of dual version of `gaussposl`: Whenever a co-Gauss formula occurring in a disjunction is found during the traversal of $\varphi$, we deduce that the whole disjunction is a co-Gauss formula. Similarly, if a conjunction consists of exclusively co-Gauss conjuncts, then the whole conjunction is a co-Gauss formula.

Algorithm `cogaussposl`$(\varphi, x, \pi)$.
Input: an $\wedge$-$\vee$-combination of atomic formulas $\varphi$, a variable $x$, a position $\pi$ in $\varphi$.
Output: a set $\{\varpi_1, \ldots, \varpi_m\}$ of pairwise independent positions such that each $\varpi_j(\varphi)$ is a co-Gauss formula; for each $j \in \{1, \ldots, m\}$ we also obtain a set of candidate solutions $\mathfrak{c}(\varpi_j)$ for the formula $\varpi_j(\varphi)$ or FAILED, which means that we were unable to compute a set of candidate solutions for the co-Gauss subformula $\varpi_j(\varphi)$.

1. If $\pi(\varphi)$ is an atomic formula $f \varrho 0$, then

   1.1. If $\varrho$ is "$\neq$," $x$ occurs in $f \in \mathbb{Z}[\mathbf{u}][x]$, and the coefficients of $f$ cannot all vanish simultaneously, then

      1.1.1. $\mathfrak{c}(\pi) :=$ `at-cs-fac`$(f \neq 0, x)$

      1.1.2. Return $\{\pi\}$.

   1.2. Return $\emptyset$.

2. For each child position $\pi|1, \ldots, \pi|n$ of $\pi$ do

   2.1. $C_i :=$ `cogaussposl`$(\varphi, x, \pi|i)$

3. Denote by $C$ the set of all co-Gauss child positions of $\pi$, i.e., $\pi|i \in C$ if and only if $C_i = \{\pi|i\}$.

4. If the top-level operator of $\pi(\varphi)$ is "$\vee$" and $|C| > 0$, then

   4.1. Select one $\pi|i \in C$ with "the best" $\mathfrak{c}(\pi|i)$.

   4.2. $\mathfrak{c}(\pi) := \mathfrak{c}(\pi|i)$

   4.3. Return $\{\pi\}$.

5. If the top-level operator of $\pi(\varphi)$ is "$\wedge$" and $|C| = n$, then

   5.1. $\mathfrak{c}(\pi) := \bigcup_{\pi|i \in C} \mathfrak{c}(\pi|i)$

   5.2. Return $\{\pi\}$.

6. Return $\bigcup_{i=1}^{n} C_i$.

At this point we would like to clarify the test "coefficient of $f$ cannot all vanish simultaneously" we use in step 1.1 of both `gaussposl` and `cogaussposl`. For atomic formula $f \varrho 0$, $f = c_k x^k + \cdots + c_1 x + c_0$, this means to decide whether the conjunction $\bigwedge_{i=0}^{k} c_i = 0$ is satisfiable. In practice this is done by incomplete methods; whenever we cannot guarantee that the conjunction is unsatisfiable we simply declare the formula $f \varrho 0$ as not being Gauss or co-Gauss, respectively, by returning $\emptyset$ in step 1.2. Our aim here is not to apply costly complete methods; in practice we rather use fast but powerful simplification techniques as described in [28], and detection of obviously unsatisfiable special cases like an occurrence of a nonzero integer coefficient $c_i$ occurring in $f$.

Next we present algorithm `atposl` that traverses the formula $\varphi$ to compute for each atomic position of $\varphi$ a set of candidate solutions. Algorithms `gaussposl` and `cogaussposl` proceed similarly, so the whole computation of Gauss, co-Gauss, and atomic positions of $\varphi$ is done in three separate traversals of the structural tree for $\varphi$, i.e., in linear time.

Algorithm `atposl`$(\varphi, x, \pi)$.
Input: an $\wedge$-$\vee$-combination of Tarski atomic formulas $\varphi$, a variable $x$, a position $\pi$ in $\varphi$.
Output: a set $\{\alpha_1, \ldots, \alpha_l\}$ of atomic positions of *all* atomic subformulas of $\varphi$ containing $x$; for each position $\alpha_j$ we also obtain a set of candidate solutions $\mathfrak{c}(\alpha_j)$ for the formula $\alpha_j(\varphi)$ or FAILED, which means that we were unable to compute a set of candidate solutions for the atomic subformula $\alpha_j(\varphi)$.

1. If $\pi(\varphi)$ is an atomic formula $f \varrho 0$, then
   - 1.1. If $f$ contains $x$, then
     - 1.1.1. $\mathfrak{c}(\pi) :=$ `at-cs-fac`$(f \varrho 0, x)$
     - 1.1.2. Return $\{\pi\}$.
   - 1.2. Return $\emptyset$.
2. For each child position $\pi|1, \ldots, \pi|n$ of $\pi$ do
   - 2.1. $A_i :=$ `atposl`$(\varphi, x, \pi|i)$
3. Return $\bigcup_{i=1}^{n} A_i$.

Now we are ready to prove the correctness of algorithm `PC-decomposition`. We begin with the following lemma, which follows directly from the definitions of finite and co-finite sets:

**Lemma 46.** *Let $F_1$ and $F_2$ be finite subsets of $\mathbb{R}$. Let $C_1$ and $C_2$ be co-finite subsets of $\mathbb{R}$, i.e, both $\mathbb{R} \setminus C_1$ and $\mathbb{R} \setminus C_2$ are finite. Let $M$ be an arbitrary subset of $\mathbb{R}$. Then the following hold:*

*(i)* $F_1 \cap M$ *is finite.*

*(ii)* $F_1 \cup F_2$ *is finite.*

*(iii)* $C_1 \cup M$ *is co-finite.*

*(iv)* $C_1 \cap C_2$ *is co-finite.* □

An indirect consequence of Lemma 46 is that a formula is *not* guaranteed to be a Gauss formula even when it consists exclusively of Gauss prime constituents. For example, the formula $\varphi$: $(u_0x - 1 = 0 \vee u_1 > 0) \wedge u_0 \geq 0$ contains only one Gauss prime at position $(1, 1)$, namely the Gauss atomic formula $u_0x - 1 = 0$. For parameter values $\mathbf{a} = (0, 1)$, however, the satisfying set $\Phi(\varphi, \mathbf{a})$ is the whole $\mathbb{R}$. The reason for this is that the satisfying set of $u_0 \geq 0$ is infinite for some parameter values.

Using Lemma 46 we now prove the correctness of the three Gauss, co-Gauss, and atomic position finding algorithms. With these lemmas at hand we then prove the correctness of `PC-decomposition`.

**Lemma 47.** *Algorithms* `gaussposl`*,* `cogaussposl`*, and* `atposl` *meet their respective specifications.*

*Proof.* We first prove the correctness of `gaussposl`. We proceed by structural induction.

Observe that the algorithm meets its specification whenever the input position $\pi$ is an atomic position: It reports an atomic formula $f \varrho 0$ to be Gauss by returning $\{\pi\}$ if and only if $\varrho$ is "=," and the coefficients of $f \in \mathbb{Z}[\mathbf{u}][x]$ cannot all vanish simultaneously. This guarantees that the solution set $\Phi(f \varrho 0, \mathbf{a})$ is finite for any parameter values $\mathbf{a} \in \mathbb{R}^m$. Furthermore, `at-cs-fac` obviously returns only "IP" test points for $f = 0$.

Assume that `gaussposl` meets its specification for each child position of $\pi$. We prove that `gaussposl` meets its specification for $\pi$. We distinguish three cases:

(a) the top-level operator is "$\wedge$" and at least one child position $\pi|i$ is Gauss: By Lemma 46 (i), the whole formula $\pi(\varphi)$ is a Gauss formula, and $\mathfrak{c}(\pi|i)$ is a set of "IP" candidate solutions for $\pi(\varphi)$ for any child position $\pi|i \in G$.

(b) the top-level operator is "$\vee$" and all child positions $\pi|i$ are Gauss: By Lemma 46 (ii), the whole formula $\pi(\varphi)$ is a Gauss formula, and the union of all candidate solutions for child formulas is a set of "IP" candidate solutions for $\pi(\varphi)$.

(c) none of the above: In this case we return all Gauss positions found in all child formulas $\pi|i(\varphi)$. The assumed correctness of `gaussposl` for all child positions $\pi|i$ ensures that $\bigcup_{i=1}^{n} G_i$ is a set of pairwise independent Gauss positions of $\pi(\varphi)$.

The proof of the correctness of `cogaussposl` using Lemma 46 (iii) and (iv) is similar, so we omit it.

Finally, algorithm `atposl` obviously meets its specification, because it merely collects all atomic formulas of $\varphi$ containing $x$ along with their positions and candidate solution sets. □

**Theorem 48.** *Algorithm* `PC-decomposition` *meets its specification.*

*Proof.* We need to prove that the positions of PCs in $P$ returned by the algorithm are pairwise independent positions that cover each atomic formula of $\varphi$ containing $x$. By Lemma 47, `atpl` computed at line 3 is a set of pairwise independent positions that cover each atomic formula of $\varphi$ containing $x$.

Since Lemma 47 ensures that the sets `gpl`, `cgpl`, and `atpl` computed at lines 1–3 are pairwise independent sets of positions of Gauss, co-Gauss, and atomic positions, the following holds: On each path from the root of $\varphi$ to any leaf of $\varphi$ there is at most one position of each of these three types. The deletions in steps 4–6 of the algorithm ensure that on each path only the first position of these is left in $\texttt{gpl} \cup \texttt{cgpl} \cup \texttt{atpl}$. Moreover, the fact that only parent or equal positions are deleted together with the fact that `atpl` already covers each atomic formula of $\varphi$ containing $x$ imply: The set of pairwise independent positions $\texttt{gpl} \cup \texttt{cgpl} \cup \texttt{atpl}$ after step 6 still covers each atomic formula of $\varphi$ containing $x$. Therefore, $P$ constructed in steps 8–10 of the algorithm is a PC decomposition of $\varphi$, which finishes the proof of the theorem. □

Theorem 48 together with Proposition 15 directly imply the following:

**Corollary 49.** *Let $P = \{\mu_1, \ldots, \mu_k\}$ be a PC decomposition of $\varphi$ computed by* `PC-decomposition`*. Denote by $\mathfrak{c}_i$ the candidate solution set of $\mu_i$, and assume that no $\mathfrak{c}_i$ is FAILED. Then $\mathfrak{c} = \bigcup_{i=1}^{k} \mathfrak{c}_i$ is a set of candidate solutions for $\varphi$.* □

At this point we would like to point to the following fact: Since Corollary 49 guarantees that the set $\mathfrak{c}$ is a set of candidate solutions for $\varphi$, we could use $\mathfrak{c}$ in the context of `vs-scheme` from Chapter 2. Instead of analyzing each atomic formula of $\varphi$ separately in step 3 of `vs-scheme`, we could first compute a PC decomposition of $\varphi$ and then extract $\mathfrak{c}$ using Corollary 49. The results of Chapter 2 guarantee that such a modified scheme is indeed correct.

**Example 50.** Here we show an example run of `PC-decomposition` on the quantifier-free formula $\varphi$ depicted in Figure 3.2. The atomic formulas there are labeled by their respective types: Gauss, co-Gauss, and other atomic formulas. `PC-decomposition` first computes sets `gpl`, `cgpl`, and `atpl` by invoking the respective algorithms.

During the call to `gaussposl`, a recursive invocation of `gaussposl` first finds a Gauss formula at atomic position $(1, 1, 1)$. Since $(1, 1)$ is an $\vee$-node, and its child $(1, 1, 2)(\varphi)$, i.e., $x > 1$ is not a Gauss formula, this is the only Gauss position returned by the algorithm, i.e., `gpl` is $\{(1, 1, 1)\}$ after step 1 of `PC-decomposition`.

Recursive invocations of `cogaussposl` find that $(1, 3, 1)$, $(1, 3, 2)$, $(2, 1)$, and $(2, 2)$ are co-Gauss positions. Since $(1, 3)$ is an $\vee$-node, and there are two co-Gauss children, the algorithm deduces that $(1, 3)$ is a co-Gauss formula; it can even decide whether candidate solutions from $x \neq 2$ or $x \neq b$ should be taken. Returning from a recursive call on $\wedge$-node (1), we obtain that (1) is not a co-Gauss formula. In contrast, returning to node (2) we see that both children of this $\wedge$-node are co-Gauss formulas, so (2) is a co-Gauss formula—with a candidate solution set obtained from *both* $x \neq 3$ and $x \neq 4$. Furthermore, at the root of the tree, which is an $\vee$-node, we have one co-Gauss child, so the whole formula is co-Gauss. Therefore, the set returned by `cogaussposl` at line 2 of `PC-decomposition` is $\{()\}$.

Algorithm `atposl` simply returns the set of all atomic positions in $\varphi$ containing $x$, i.e.,

$$\big\{(1, 1, 1), (1, 1, 2), (1, 2), (1, 3, 1), (1, 3, 2), (2, 1), (2, 2)\big\}.$$

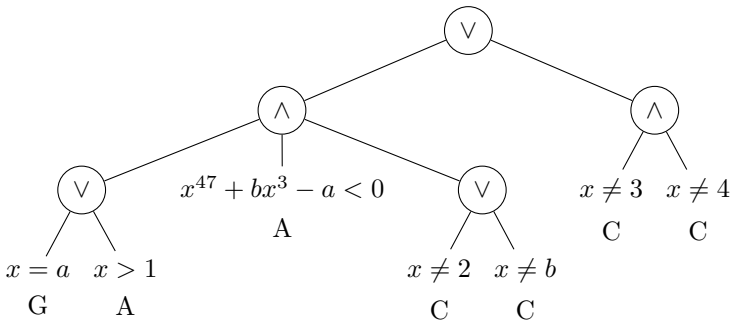



Figure 3.2: An example input formula $\varphi$ for `PC-decomposition` discussed in Example 50. Leaf positions are labeled as "G" (Gauss), "C" (co-Gauss), and "A" (other atomic formulas).

Notice, however, that the candidate solution set for the prime constituent at position $(1, 2)$ is FAILED, because of the high-degree irreducible polynomial $x^{47} + bx^3 - a$.

After step 6 of `PC-decomposition` the sets `gpl` and `atpl` are empty, and `cgpl` is $\{()\}$ with candidate solution set obtained by taking candidate solution sets for formulas $x \neq 3$ and $x \neq 4$. The remaining steps of `PC-decomposition` merely compute a set of prime constituents from this position and candidate solution set.

To conclude the discussion of this example we point to two advantages of `PC-decomposition` it demonstrates: First, an atomic formula that would normally cause a failure of the virtual substitution approach can be omitted, and we do not need to compute candidate solutions for it at all. Second, even when other atomic formulas are not critical, we see that they can be found redundant: Notice that from the seven atomic formulas in $\varphi$ we used only two for the generation of candidate solutions and prime constituents. ◇

We conclude this section with a few remarks on the usefulness of PC decompositions and similarities between them and CAD. First recall that the notion of a prime constituent is a generalization of the notion of atomic formula. A prime constituent decomposition is an efficient tool used primarily to take advantage of equations and negated equations occurring in $\varphi$. Indeed, Gauss and co-Gauss prime constituents will play a special role in the quantifier elimination algorithm scheme presented in the upcoming sections.

For Gauss prime constituents we will benefit from the finiteness of their satisfying sets for any parameter values $\mathbf{a} \in \mathbb{R}^m$. The intuition is that after substituting the finitely many test points generated by a Gauss prime constituent into $\varphi$ we can replace the Gauss subformula with "false" while substituting the other test points. The reason for this is that whenever this particular Gauss subformula has to be true for $\varphi$ to be true, there are only finitely many choices how the Gauss prime constituent can be made true—and these choices are described by its candidate solutions that generated "Gauss" test points afterwards.

For co-Gauss prime constituents, in contrast, this intuition is not correct. Therefore, we cannot simply replace a co-Gauss subformula with "false" after substituting its candidate solutions into $\varphi$. Nevertheless, a co-Gauss prime

constituent containing more complex subformulas can allow us to ignore parts of the formula during the test point generation process—as we have seen in Example 50.

Thank to these properties, we can view the Gauss and co-Gauss prime constituents of $\varphi$ as a kind of equational and negated equational constraints for $\varphi$ and its subformulas, respectively. Equational constraints—which are equations implied by $\varphi$—are a well-established concept that allows one to optimize a CAD construction in two ways: First by omitting some resultant computations during the CAD projection phase, and afterwards during the lifting phase to lift only w.r.t. equational constraints [53, 54, 33]. Even though we look primarily at the Boolean structure of $\varphi$ and not at some equational constraint of the whole formula $\varphi$, the facts that a Gauss prime constituent prevents some atomic formulas from generating test points, or that it can be replaced with "false" during substitution of other test points are similarities that resemble the avoidance of some resultant computations and root isolations in the CAD world. Therefore, it seems to be a promising research direction to investigate possible connections between our Gauss and co-Gauss prime constituents on one side and equational constraints in the CAD context on the other.

## 3.2 Conjunctive Associativity and Marking

Let $\pi_1$ and $\pi_2$ be two independent positions in $\varphi$. Let $\pi = \operatorname{lca}(\pi_1, \pi_2)$. If $\pi$ identifies an $\wedge$-node in the structural tree for $\varphi$, i.e., the top-level Boolean operator of $\pi(\varphi)$ is "$\wedge$," then the positions $\pi_1$ and $\pi_2$ are said to be *conjunctively associated*. The conjunctive associativity definition excludes dependent (and in particular equal) positions $\pi_1$ and $\pi_2$. Observe that the formulas $\pi_1(\varphi)$ and $\pi_2(\varphi)$ can be equal Tarski formulas even when $\pi_1$ is by definition different from position $\pi_2$.

**Example 51.** Consider a quantifier-free Tarski formula $\varphi$:

$$\big((x - b = 0 \wedge x - a \leq 0) \vee x - 1 > 0\big) \wedge x < 0 \wedge \big(x - 2 \neq 0 \vee x - a \leq 0\big).$$

In Figure 3.3 we can see the structural tree for $\varphi$. Observe that (2) is conjunctively associated with $(1, 2)$. On the other hand, $()$ is *not* conjunctively associated with $(1, 1, 1)$ even though $\operatorname{lca}\big((), (1, 1, 1)\big) = ()$ is an $\wedge$-node. The reason is that the positions $(1, 1, 1)$ and $()$ are not independent, more specifically $()$ is a parent position of the atomic position $(1, 1, 1)$, formally $(1, 1, 1) \sqsubset ()$. Similarly, $(3, 1)$ and $(3, 2)$ are not conjunctively associated, because they are independent and their lowest common ancestor $()$ is an $\vee$-node. $\diamond$

In the following we denote the fact that $\pi_1$ and $\pi_2$ are conjunctively associated positions in $\varphi$ by $\pi_1 \curlywedge_\varphi \pi_2$. Observe that the binary relation $\curlywedge_\varphi \subseteq \mathtt{Pos}(\varphi) \times \mathtt{Pos}(\varphi)$ of conjunctive associativity on positions is:

(i) anti-reflexive: Since $\pi_1$ and $\pi_2$ need to be independent to be conjunctively associated, $\pi_1 \curlywedge_\varphi \pi_2$ implies $\pi_1 \neq \pi_2$.

(ii) symmetric: $\pi_1 \curlywedge_\varphi \pi_2$ whenever $\pi_2 \curlywedge_\varphi \pi_1$, because $\operatorname{lca}(\pi_1, \pi_2) = \operatorname{lca}(\pi_2, \pi_1)$.

(iii) not always transitive: As we can see from Figure 3.3, $(1, 1, 1) \curlywedge_\varphi (2)$ and $(2) \curlywedge_\varphi (1, 2)$. At the same time, $(1, 1, 1)$ is *not* conjunctively associated

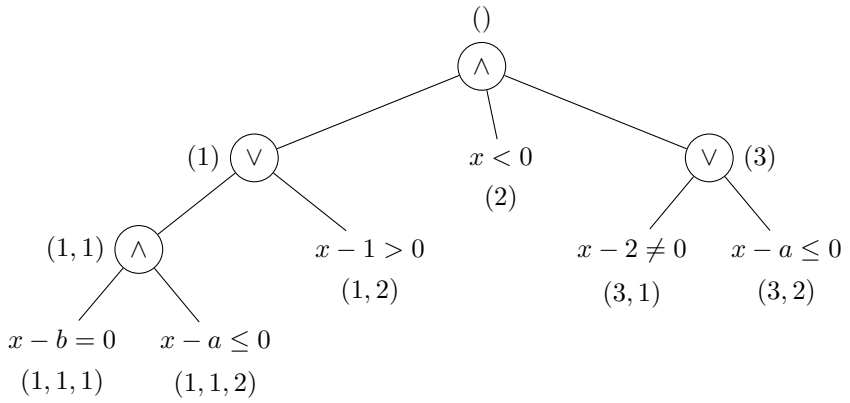



Figure 3.3: The structural tree for a quantifier-free Tarski formula.

with $(1,2)$. Similarly, $(3,1) \curlywedge_\varphi (2)$ and $(2) \curlywedge_\varphi (3,2)$, but $(3,1) \curlywedge_\varphi (3,2)$ does *not* hold. It is not hard to see that $\curlywedge_\varphi$ is transitive whenever $\varphi$ does not contain positions $\pi_1$, $\pi_2$ such that $\pi_1 \sqsubset \pi_2$ and $\pi_1$ is an $\vee$-node and $\pi_2$ is an $\wedge$-node.

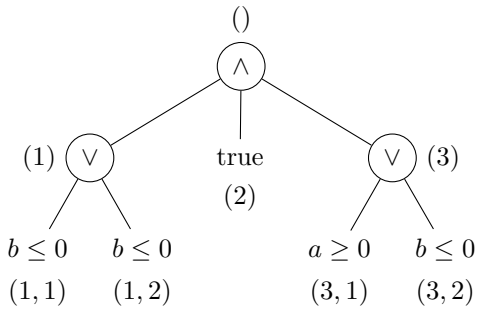
In the following we will simply write $\pi_1 \curlywedge \pi_2$ instead of $\pi_1 \curlywedge_\varphi \pi_2$ whenever the formula $\varphi$ will be clear from the context.

Our aim in the following is to prove that a pair of atomic positions in $\varphi$ is conjunctively associated if and only if there exists a disjunct of a particular disjunctive normal form of $\varphi$ containing the atomic formulas at both positions.

To begin with, we precisely define "the DNF" of $\varphi$: Whenever we say "the DNF" of an $\wedge$-$\vee$-combination of atoms $\varphi$, we mean the output of Algorithm `the-dnf`$(\varphi, ())$ given below. We emphasize that our aim here is *not* to compute the DNF of $\varphi$ during the quantifier elimination of $\exists x$ from $\exists x(\varphi)$. We use the DNF only to prove properties of the conjunctive associativity relation $\curlywedge_\varphi$. Of course we will use these properties in later sections to prove the correctness of our structural quantifier elimination scheme given there.

Algorithm `the-dnf`$(\varphi, \pi)$.
Input: an $\wedge$-$\vee$-combination of atoms $\varphi$, a position $\pi \in$ `Pos`$(\varphi)$.
Output: a nonempty multiset $L = \{C_1, \ldots, C_n\}$ of formulas such that each $C_i$ is an atomic formula or a conjunction of atomic formulas, and $\pi(\varphi) \longleftrightarrow \bigvee_{i=1}^{n} C_i$.

1. If $\pi(\varphi)$ is an atomic formula, then return $\{\pi(\varphi)\}$.
2. For each child position $\pi|1, \ldots, \pi|k$ of $\pi$ do
   2.1. $L_j :=$ `the-dnf`$(\varphi, \pi|j)$
   2.2. $n_j := |L_j|$
3. If the top-level operator of $\pi(\varphi)$ is "$\vee$," then
   3.1. Return $\biguplus_{j=1}^{k} L_j$.
4. If the top-level operator of $\pi(\varphi)$ is "$\wedge$," then




| $\pi$ | $\texttt{the-dnf}(\varphi, \pi)$ |
|---|---|
| $(1,1)$ | $\{b \leq 0\}$ |
| $(1,2)$ | $\{b \leq 0\}$ |
| $(1)$ | $\{b \leq 0, b \leq 0\}$ |
| $(2)$ | $\{\text{true}\}$ |
| $(3,1)$ | $\{a \geq 0\}$ |
| $(3,2)$ | $\{b \leq 0\}$ |
| $(3)$ | $\{a \geq 0, b \leq 0\}$ |
| $()$ | $\{b \leq 0 \wedge \text{true} \wedge a \geq 0, b \leq 0 \wedge \text{true} \wedge b \leq 0,$ $b \leq 0 \wedge \text{true} \wedge a \geq 0, b \leq 0 \wedge \text{true} \wedge b \leq 0\}$ |

Figure 3.4: An $\wedge$-$\vee$-combination $\varphi$ with the output of $\texttt{the-dnf}(\varphi, \pi)$ for every position $\pi \in \texttt{Pos}(\varphi)$.

4.1. Denote the product $\{1, \ldots, n_1\} \times \cdots \times \{1, \ldots, n_k\}$ by $M$.

4.2. Return $\left\{\bigwedge_{j=1}^{k} C_{m_j} \mid (m_1, \ldots, m_k) \in M \text{ and } C_{m_j} \in L_j\right\}$.

It is not hard to see that Algorithm `the-dnf` meets its specification, and that $\texttt{the-dnf}(\varphi, \pi) = \texttt{the-dnf}(\pi(\varphi), ())$. The output of the algorithm for a simple input formula $\varphi$ is for illustration shown in Figure 3.4.

It is well-known that the length of the DNF of $\varphi$ can indeed be exponential in the length of $\varphi$. Nevertheless, we prove a precise formulation of this result for completeness here, because we will use it later in this thesis:

**Proposition 52.** *Let $\varphi$ be a Tarski formula meeting the following specification:*

1. *The structural tree for $\varphi$ is a full binary tree with height $2h$, i.e., each inner node has exactly two children, and $\varphi$ has $2^{2h}$ atomic positions.*

2. *An inner node with an odd distance from the root is an $\vee$-node.*

3. *An inner node with an even distance from the root is an $\wedge$-node.*

*Let $L = \{C_1, \ldots, C_n\}$ be the DNF computed by $\texttt{the-dnf}(\varphi, ())$. Then the following hold:*

*(i) The number $n$ of the members of the DNF of $\varphi$ is $2^{2^{h+1}-2}$.*

*(ii) Each formula $C_i$ is a conjunction of $2^h$ atomic formulas occurring in $\varphi$.*

*Proof.* (i) Denote by $T_{2k}$ the number of DNF members of the DNF computed by `the-dnf`$(\psi_{2k}, ())$ for a Tarski formula $\psi_{2k}$ with height $2k$ meeting the specification 1–3. The specification ensures that $\psi_{2k}$ is of the following form:

$$\big(\psi^{1,1}_{2(k-1)} \vee \psi^{1,2}_{2(k-1)}\big) \wedge \big(\psi^{2,1}_{2(k-1)} \vee \psi^{2,2}_{2(k-1)}\big),$$

where $\psi^{1,1}_{2(k-1)}$, $\psi^{1,2}_{2(k-1)}$, $\psi^{2,1}_{2(k-1)}$, and $\psi^{2,2}_{2(k-1)}$ are formulas with height $2(k-1)$ meeting the specification 1–3. Looking at algorithm `the-dnf` we can conclude that the following recurrence relation holds for $T_{2k}$:

$$\begin{aligned} T_0 &= 1, \\ T_{2k} &= (2 \cdot T_{2(k-1)})^2 \quad \text{if } k > 0. \end{aligned}$$

We prove by induction on $k$ that $T_{2k} = 2^{2^{k+1}-2}$ is the solution of this recurrence: If $k = 0$, then $T_0 = 1$. At the same time, $2^{2^{k+1}-2} = 2^{2^1-2} = 1$. Let now $k > 0$ and assume that the identity holds for $k-1$. Expanding the recurrence relation and using the induction hypothesis we obtain that

$$T_{2k} = (2 \cdot T_{2(k-1)})^2 = (2 \cdot 2^{2^k-2})^2 = 2^{2^{k+1}-2}.$$

This shows that $T_{2k} = 2^{2^{k+1}-2}$ for every $k \geq 0$, which proves (i).

(ii) Denote by $S_{2k}$ the number of atomic formulas in each DNF member $C_i$ computed by `the-dnf`$(\psi_{2k}, ())$ for a Tarski formula $\psi_{2k}$ with height $2k$ meeting the specification 1–3. Similarly as in the proof of (i), we deduce that the following recurrence relation holds for $S_{2k}$:

$$\begin{aligned} S_0 &= 1, \\ S_{2k} &= 2 \cdot S_{2(k-1)} \quad \text{if } k > 0. \end{aligned}$$

From this recurrence it is clear that $S_{2k} = 2^k$, which proves (ii). □

Having defined "the DNF" of $\varphi$, we now continue with the notions of condensing and deletion.

Let $\pi$ be a position in $\varphi$. The *condensing of $\varphi$ with respect to $\pi$* is the formula $\Gamma(\varphi, \pi)$ obtained by replacing all atomic formulas of $\varphi$ at positions independent from and not conjunctively associated with $\pi$ with "false." For example, for formula $\varphi$ in Figure 3.3 we have $\Gamma\big(\varphi, (1,1)\big) = \Gamma\big(\varphi, (1,1,1)\big) = \Gamma\big(\varphi, (1,1,2)\big)$, which all yield

$$\big((x - b = 0 \wedge x - a \leq 0) \vee \text{false}\big) \wedge x < 0 \wedge \big(x - 2 \neq 0 \vee x - a \leq 0\big),$$

and $\Gamma\big(\varphi, ()\big) = \Gamma\big(\varphi, (1)\big) = \Gamma\big(\varphi, (2)\big) = \Gamma\big(\varphi, (3)\big)$, which all yield $\varphi$.

Let $\pi_1, \ldots, \pi_n$ be $n$ positions in $\varphi$. The *deletion of $\pi_1, \ldots, \pi_n$ from $\varphi$* is the formula $\Delta(\varphi, \{\pi_1, \ldots, \pi_n\})$ obtained by replacing all atomic subformulas of the $n$ formulas $\pi_1(\varphi)$, $\ldots$, $\pi_n(\varphi)$ in $\varphi$ with "false." Since a replacement of atomic formulas in $\varphi$ with "false" does not change the structural tree of the whole formula $\varphi$, the order in which these $n$ replacements take place is unimportant. For example, using again formula $\varphi$ from Figure 3.3, $\Delta\big(\varphi, \{()\}\big)$ yields

$$\big((\text{false} \wedge \text{false}) \vee \text{false}\big) \wedge \text{false} \wedge \big(\text{false} \vee \text{false}\big),$$

$\Delta\big(\varphi, \{(1), (3)\}\big)$ yields

$$\big((\text{false} \wedge \text{false}) \vee \text{false}\big) \wedge x < 0 \wedge \big(\text{false} \vee \text{false}\big),$$

and $\Delta\big(\varphi, \{(1,1,1), (1,1), (2), (3,2)\}\big)$ yields

$$\big((\text{false} \wedge \text{false}) \vee x - 1 > 0\big) \wedge \text{false} \wedge \big(x - 2 \neq 0 \vee \text{false}\big).$$

Let $\pi_1, \ldots, \pi_n$, and $\pi$ be $n+1$ positions in $\varphi$. In the following we denote $\Delta(\varphi, \{\pi_1, \ldots, \pi_n\})$ by $\varphi^{\{\pi_1,\ldots,\pi_n\}}$ and $\Gamma(\varphi, \pi)$ by $\varphi_\pi$. Obviously, $\varphi^\emptyset = \varphi$. Furthermore, we denote $\Gamma(\varphi^{\{\pi_1,\ldots,\pi_n\}}, \pi)$ by $\varphi_\pi^{\{\pi_1,\ldots,\pi_n\}}$, i.e., we formally first apply the deletion operator and then the condensing operator. Observe, however, that the replacement order does not play any role here either, because both operators leave the structure of a structural tree intact.

Denote by $\mathtt{Sf}(\varphi)$ the set of all subformulas of $\varphi$. Denote by $\mathtt{At}(\varphi) \subseteq \mathtt{Sf}(\varphi)$ the set of all atomic subformulas of $\varphi$. The *conjunctive associativity relation* $\curlywedge_\varphi^{\mathrm{SF}} \subseteq \mathtt{Sf}(\varphi) \times \mathtt{Sf}(\varphi)$ *on subformulas* is defined as follows: $\alpha_1 \curlywedge_\varphi^{\mathrm{SF}} \alpha_2$ if there exist two conjunctively associated positions $\pi_1, \pi_2 \in \mathtt{Pos}(\varphi)$ such that $\pi_1(\varphi) = \alpha_1$ and $\pi_2(\varphi) = \alpha_2$.

Similarly as with the conjunctive associativity relation $\curlywedge_\varphi$ on positions, the conjunctive associativity relation $\curlywedge_\varphi^{\mathrm{SF}}$ on subformulas is symmetric but not necessarily transitive. In contrast to $\curlywedge_\varphi$, the relation $\curlywedge_\varphi^{\mathrm{SF}}$ is not necessarily antireflexive, because a subformula can occur in $\varphi$ at two different conjunctively associated positions. For example, we have $x - a \leq 0 \curlywedge_\varphi^{\mathrm{SF}} x - a \leq 0$ for formula $\varphi$ from Figure 3.3, because $(3,2)(\varphi) = (1,1,2)(\varphi) = x - a \leq 0$, and $(3,2) \curlywedge_\varphi (1,1,2)$.

At this point we change our perspective: From "the Tarski world" we switch to "the Boolean world" for a while. Note that the definitions of $\mathtt{Sf}$, $\mathtt{At}$, conjunctive associativity relation on positions and on subformulas carry over to the Boolean world. Our aim is to investigate some properties that are completely independent of the Tarski world, i.e., already the Boolean structure of a formula ensures their validity. Afterwards we will indeed apply these properties to quantifier-free Tarski formulas.

**Proposition 53.** *Let $\varphi$ be an $\wedge$-$\vee$-combination of Boolean atoms. Let $\pi_1$, $\pi_2 \in \mathtt{Pos}(\varphi)$ and $D_1$, $D_2 \subseteq \mathtt{Pos}(\varphi)$ be such that $\pi_1 \sqsubseteq \pi_2$ and $D_2 \subseteq D_1$. Then $\varphi_{\pi_1}^{D_1} \longrightarrow \varphi_{\pi_2}^{D_2}$.*

*Proof.* Assume that $\varphi_{\pi_1}^{D_1}$ holds. We have to prove that $\varphi_{\pi_2}^{D_2}$ holds as well.

Let $X_i$ be the set of all atomic positions of $\varphi$ that are independent from and not conjunctively associated with $\pi_i$. Since $\pi_1 \sqsubseteq \pi_2$, i.e., $\pi_2$ is not deeper in $\varphi$ than $\pi_1$, we conclude that $X_2 \subseteq X_1$. Similarly, let $Y_i$ be the set of all atomic positions of $\varphi$ that are descendants of some position in the set $D_i$. Since $D_2 \subseteq D_1$, we obviously have $Y_2 \subseteq Y_1$.

Observe that formula $\varphi_{\pi_i}^{D_i}$ is obtained from $\varphi$ by replacing atomic formulas at positions $X_i \cup Y_i$ with "false." Since $(X_2 \cup Y_2) \subseteq (X_1 \cup Y_1)$, we deduce that $\varphi_{\pi_1}^{D_1}$ can be obtained from $\varphi_{\pi_2}^{D_2}$ by replacing atomic formulas at positions $(X_1 \cup Y_1) \setminus (X_2 \cup Y_2)$ in formula $\varphi_{\pi_2}^{D_2}$ with "false." We assume that $\varphi_{\pi_1}^{D_1}$ holds, so the fact that $\varphi$ is an $\wedge$-$\vee$-combination of atomic formulas ensures that $\varphi_{\pi_2}^{D_2}$ holds as well. $\square$

The following lemma follows directly from the definition of the conjunctive associativity relation on subformulas:

**Lemma 54.** *Let $\varphi$ be an $\wedge$-$\vee$-combination of Boolean atoms. Let $\varphi' \in \mathtt{Sf}(\varphi)$ and let $\varphi_1, \varphi_2 \in \mathtt{Sf}(\varphi')$. Assume that $\varphi_1 \curlywedge^{\mathrm{SF}}_{\varphi'} \varphi_2$. Then $\varphi_1 \curlywedge^{\mathrm{SF}}_{\varphi} \varphi_2$.* □

Now we are finally ready to prove a characterization of conjunctive associativity relation $\curlywedge^{\mathrm{SF}}_{\varphi}$ on subformulas based on the DNF of $\varphi$.

**Theorem 55.** *Let $\varphi$ be an $\wedge$-$\vee$-combination of Boolean atoms. Let $\pi \in \mathtt{Pos}(\varphi)$, and let $L = \{C_1, \ldots, C_n\}$ be the output of* $\mathtt{the\text{-}dnf}(\varphi, \pi)$. *Let $\alpha_1, \alpha_2 \in \mathtt{At}(\varphi)$ be two (not necessarily distinct) atomic formulas. Then $\alpha_1 \curlywedge^{\mathrm{SF}}_{\varphi} \alpha_2$ if and only if there exists $C_i \in L$ such that $\alpha_1$ and $\alpha_2$ occur at two distinct positions in $C_i$.*

*Proof.* We proceed by structural induction on the structure of the position $\pi$. To begin with, observe that the theorem holds whenever $\pi$ is an atomic position of $\varphi$: In that case we have $\curlywedge^{\mathrm{SF}}_{\pi(\varphi)} = \emptyset$, and $\mathtt{the\text{-}dnf}(\varphi, \pi) = \{\pi(\varphi)\}$, so the equivalence asserted by the theorem obviously holds.

Let now $\pi$ be an inner position in $\varphi$ with $k$ children. Assume that the theorem holds for every child position $\pi|j$, $j \in \{1, \ldots, k\}$. For each $j \in \{1, \ldots, k\}$ denote by $L_j$ the result of $\mathtt{the\text{-}dnf}(\varphi, \pi|j)$ and denote $|L_j|$ by $n_j$. We prove that $\alpha_1 \curlywedge^{\mathrm{SF}}_{\pi(\varphi)} \alpha_2$ if and only if there exists $C_i \in L$ such that $\alpha_1$ and $\alpha_2$ occur at two distinct positions in $C_i$. We distinguish two cases:

1. the top-level operator of $\pi(\varphi)$ is "$\vee$:" In this case we have $L = \biguplus_{j=1}^{k} L_j$.

   Assume first that $\alpha_1 \curlywedge^{\mathrm{SF}}_{\pi(\varphi)} \alpha_2$. Then the lowest common ancestor $\lambda$ of some two conjunctively associated positions of $\alpha_1$ and $\alpha_2$ is an $\wedge$-node. Thus, $\lambda \sqsubseteq \pi|j$ for some $j \in \{1, \ldots, k\}$. Therefore, $\alpha_1 \curlywedge^{\mathrm{SF}}_{\pi|j(\varphi)} \alpha_2$, and the induction hypothesis ensures that $\alpha_1$ and $\alpha_2$ occur at two distinct positions in $C_i$ for some $C_i \in L_j \subseteq L$.

   Assume now that $\alpha_1$ and $\alpha_2$ occur at two distinct positions in $C_i$ for some $C_i \in L$. Since $L = \biguplus_{j=1}^{k} L_j$, $\alpha_1$ and $\alpha_2$ occur in some $C_i \in L_j$ for some $j \in \{1, \ldots, k\}$. The induction hypothesis therefore ensures that $\alpha_1 \curlywedge^{\mathrm{SF}}_{\pi|j(\varphi)} \alpha_2$. Using Lemma 54 we obtain that $\alpha_1 \curlywedge^{\mathrm{SF}}_{\pi(\varphi)} \alpha_2$.

2. the top-level operator of $\pi(\varphi)$ is "$\wedge$:" In this case we have $L = \big\{ \bigwedge_{j=1}^{k} C_{m_j} \mid (m_1, \ldots, m_k) \in M \big\}$, where $M = \{1, \ldots, n_1\} \times \cdots \times \{1, \ldots, n_k\}$.

   Assume first that $\alpha_1 \curlywedge^{\mathrm{SF}}_{\pi(\varphi)} \alpha_2$. Denote by $\lambda$ the lowest common ancestor of some two conjunctively associated positions of $\alpha_1$ and $\alpha_2$. We distinguish two cases:

   (a) If $\lambda \neq \pi$, then the induction hypothesis ensures that $\alpha_1$ and $\alpha_2$ occur at two distinct positions in some $C_{m_j} \in L_j$ for some $j \in \{1, \ldots, k\}$ and $m_j \in \{1, \ldots, n_j\}$. Thus, the construction of $L$ ensures that $\alpha_1$ and $\alpha_2$ indeed occur at two distinct positions in some $\bigwedge_{j=1}^{k} C_{m_j} \in L$ as well.

   (b) If $\lambda = \pi$, then w.l.o.g. $\alpha_1$ occurs at a position in $C_{m_1}$ and $\alpha_2$ occurs at a position in some $C_{m_2}$, where $m_1 \in \{1, \ldots, n_1\}$ and $m_2 \in \{1, \ldots, n_2\}$. The construction of $L$ ensures that $\alpha_1$ and $\alpha_2$ occur at two distinct positions in some $\bigwedge_{j=1}^{k} C_{m_j} \in L$ as well.

Assume now that $\alpha_1$ and $\alpha_2$ occur at two distinct positions in some $\bigwedge_{j=1}^{k} C_{m_j} \in L$. W.l.o.g. there are two cases to consider:

(a) If both $\alpha_1$ and $\alpha_2$ occur in some $C_{m_j} \in L_j$ for some $j \in \{1, \ldots, k\}$ and $m_j \in \{1, \ldots, n_j\}$, then the induction hypothesis ensures that $\alpha_1 \curlywedge^{\mathrm{SF}}_{\pi(\varphi)|j} \alpha_2$. Using Lemma 54 we obtain that $\alpha_1 \curlywedge^{\mathrm{SF}}_{\pi(\varphi)} \alpha_2$.

(b) The case when $\alpha_1$ occurs in some $C_{m_1}$ and $\alpha_2$ occurs in some $C_{m_2}$, where $m_1 \in \{1, \ldots, n_1\}$ and $m_2 \in \{1, \ldots, n_2\}$. This means that the lowest common ancestor of the positions of $\alpha_1$ and $\alpha_2$ is $\pi$, which is an $\wedge$-node, because we assume that the top-level operator of $\pi(\varphi)$ is "$\wedge$." This means, by definition, that $\alpha_1$ and $\alpha_2$ are conjunctively associated in $\pi(\varphi)$, i.e., $\alpha_1 \curlywedge^{\mathrm{SF}}_{\pi(\varphi)} \alpha_2$. □

Since `the-dnf` does not delete any duplicates in its output formulas $C_i$, we can use Theorem 55 to deduce: An atomic formula $\alpha$ occurring in $\varphi$ is conjunctively associated with itself, formally $\alpha \curlywedge^{\mathrm{SF}}_{\varphi} \alpha$, if and only if there exists $C_i \in L$ containing $\alpha$ at least twice.

As the final result of this section we present the so-called Marking technique. The technique uses condensing and deletion to obtain an equivalent reformulation of $\varphi$. Informally, it can be described as follows: Mark $n$ arbitrary atoms in an $\wedge$-$\vee$-combination of Boolean atoms $\varphi$. Then condense w.r.t. each of the marked atoms. Afterwards replace in $\varphi$ each marked formula with "false." The disjunction of the $n$ condensed formulas and of the "rest after the deletion" is then equivalent to the original formula:

**Theorem 56** (Marking)**.** *Let $\varphi$ be an $\wedge$-$\vee$-combination of pairwise distinct Boolean atoms $\alpha_1, \ldots, \alpha_m$. Denote by $\pi_1, \ldots, \pi_m$ the positions of these atoms, respectively. Let $n \in \{1, \ldots, m\}$. Let $D \subseteq \{\pi_1, \ldots, \pi_n\}$. Then the following equivalence holds:*

$$\varphi \longleftrightarrow \varphi_{\pi_1} \vee \cdots \vee \varphi_{\pi_n} \vee \varphi^D.$$

*Proof.* To begin with, we show that each formula from $\{\varphi_{\pi_1}, \ldots, \varphi_{\pi_n}, \varphi^D\}$ implies $\varphi$. The formula $\varphi$ is an $\wedge$-$\vee$-combination of atoms, and each of the formulas $\{\varphi_{\pi_1}, \ldots, \varphi_{\pi_n}, \varphi^D\}$ was obtained from $\varphi$ by replacing some of its subformulas with "false." Thus, the monotonicity of $\varphi$ ensures that if at least one of the formulas from $\{\varphi_{\pi_1}, \ldots, \varphi_{\pi_n}, \varphi^D\}$ holds, then $\varphi$ holds as well.

Now assume that $\varphi$ holds. We show that $\varphi_{\pi_1} \vee \cdots \vee \varphi_{\pi_n} \vee \varphi^D$ holds as well. Since $\varphi$ holds, a formula $C_i \in \texttt{the-dnf}(\varphi, ())$ holds as well. Without loss of generality we can assume that the formula $C_i$ is of the form $\alpha_{i,1} \wedge \cdots \wedge \alpha_{i,k}$ for some $k \in \{1, \ldots, m\}$. In other words, if the atoms $\alpha_{i,1}, \ldots, \alpha_{i,k}$ hold, then the formula $\varphi$ holds as well. We distinguish two cases:

(a) There exists $j \in \{1, \ldots, n\}$ such that $\alpha_j$ occurs in $C_i$: By Theorem 55, the atoms $\alpha_{i,1}, \ldots, \alpha_{i,k}$ occurring in $C_i$ are pairwise conjunctively associated in $\varphi$. Since we assume that the Boolean atoms in $\varphi$ are pairwise distinct, and the condensing operator never replaces conjunctively associated formulas with "false," we deduce that none of the atoms $\alpha_{i,1}, \ldots, \alpha_{i,k}$ occurring in $C_i$ was replaced with "false" to obtain $\varphi_{\pi_j}$ from $\varphi$. The fact that $C_i$ holds together with the monotonicity of $\varphi$ ensure that $\varphi_{\pi_j}$ holds.

(b) For all $j \in \{1, \ldots, n\}$ we have that $\alpha_j$ does not occur in $C_i$: Observe that $\varphi^D$ is obtained from $\varphi$ by replacing some of its atomic subformulas

at positions $\pi_1, \ldots, \pi_n$ with "false." Since we assume that the Boolean atoms are pairwise distinct, none of the atoms $\alpha_{i,1}, \ldots, \alpha_{i,k}$ occurring in $C_i$ was replaced with "false" to obtain $\varphi^D$. The fact that $C_i$ holds together with the monotonicity of $\varphi$ thus ensure that $\varphi^D$ holds.

In both cases we have proven that a formula from $\{\varphi_{\pi_1}, \ldots, \varphi_{\pi_n}, \varphi^D\}$ holds. This finishes the proof of the theorem. □

Here we return to "the Tarski world." It is not hard to see that Proposition 53, Lemma 54, and Theorem 55 remain valid when we replace "Boolean atom" with "Tarski atomic formula" in their statements. Furthermore, Theorem 56, which we will use in the next section, remains valid even when we substitute for the pairwise distinct Boolean atoms *arbitrary* quantifier-free Tarski formulas. We conclude this section with a simple example of this approach:

**Example 57.** Consider the Tarski formula $\varphi$ in Figure 3.5 with two Gauss prime constituents $G_1$ and $G_2$ at positions $\pi_1$ and $\pi_2$, respectively. Treating the subformulas $G_1$, $G_2$, $x + 1 < 0$, and $x - 1 > 0$ as pairwise distinct Boolean atoms and applying Marking (Theorem 56) to $\pi_1$ and $\pi_2$ leads to an equivalent of $\varphi$ shown in Figure 3.6. At the same time, it is possible to first apply Marking to position $\pi_1$ and afterwards to position $\pi_2$ in $\varphi^{\{\pi_1\}}$, because $\pi_1$ and $\pi_2$ are independent. This leads to formula in Figure 3.7. For the sake of simplicity we made obvious simplifications involving "false" in Figure 3.6 and Figure 3.7. To sum up, Theorem 56 ensures that $\varphi_{\pi_1} \vee \varphi_{\pi_2} \vee \varphi^{\{\pi_1,\pi_2\}}$ and $\varphi_{\pi_1} \vee \varphi_{\pi_2}^{\{\pi_1\}} \vee \varphi^{\{\pi_1,\pi_2\}}$ are equivalent to $\varphi$. ◇

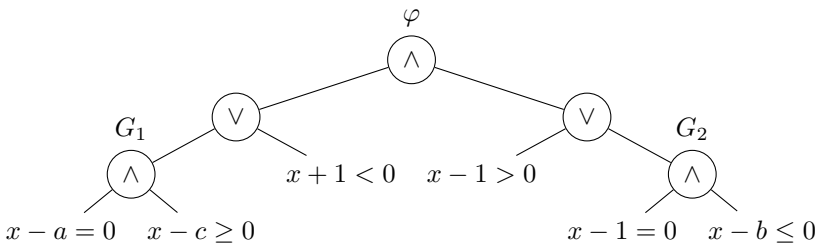



Figure 3.5: Formula $\varphi$ from Example 57.

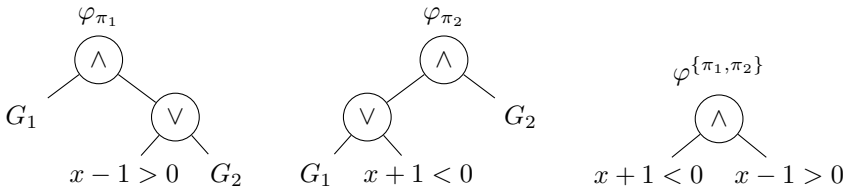



Figure 3.6: The result of Marking applied to $\pi_1$ and $\pi_2$ in $\varphi$.

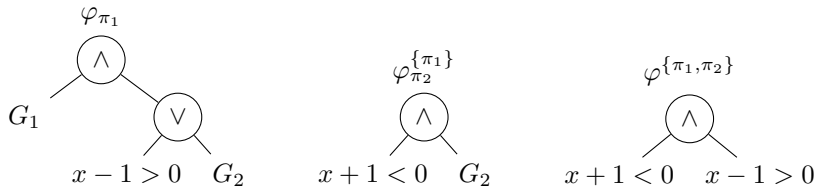



Figure 3.7: The result of Marking applied to $\pi_1$ in $\varphi$ and $\pi_2$ in $\varphi^{\{\pi_1\}}$.

## 3.3 Structural QE Algorithm Scheme

In Section 3.1 we have shown how to compute a prime constituent decomposition of $\varphi$. In Corollary 49 we have shown that a PC decomposition of $\varphi$ gives rise to a set of candidate solutions for $\varphi$. Such a set of candidate solutions can indeed be directly used in the context of `vs-scheme` of Chapter 2.

In this section we show a more clever use of a PC decomposition of $\varphi$. We present an enhanced QE algorithm scheme `svs-scheme` that takes advantage of the positional origin and structural information stored in prime constituents of the input formula $\varphi$.

To begin with, we turn to the definition of a structural test point. This notion generalizes its corresponding "non-structural" variant introduced in Chapter 2 to allow for tracing the structural origin of such a "non-structural" test point. A *structural test point* is a triple $(e, \pi, \mathcal{F})$ with the following properties:

(i) $e$ is a test point as defined in Chapter 2, i.e., $e$ is of the form $(f, S)$, $(f, S) \pm \varepsilon$, where $(f, S)$ is a parametric root description, or $\pm\infty$,

(ii) $\pi$ is a position in the input formula $\varphi$, and

(iii) $\mathcal{F}$ is a set of pairwise independent positions in $\varphi$.

In Chapter 2 we defined an elimination set for $\varphi$ and $x$ as a set of test points that when substituted into $\varphi$ by means of virtual substitution, yield a quantifier-free equivalent of $\exists x(\varphi)$. Here we proceed in a similar way and define a *structural elimination set $E$ for $\varphi$ and $x$* as a finite set of structural test points that yield a quantifier-free equivalent of $\exists x(\varphi)$ when substituted into $\varphi$. To exploit the structural information of $E$, however, we will use *structural virtual substitution*, which substitutes $(e, \pi, \mathcal{F})$ into $\varphi_{\varphi}^{\mathcal{F}}$. This contrasts with the "usual" virtual substitution of Chapter 2, which substitutes a test point $e$ into the whole formula $\varphi$. In this way we replace all atomic formulas at positions lying at or under the false replacement positions $\mathcal{F}$ and at positions independent from and not conjunctively associated with $\pi$ with "false" instead of applying a costly virtual substitution $[x /\!\!/ e]$ to them.

Before presenting the enhanced QE algorithm scheme we first show how to turn a PC decomposition of $\varphi$ into a set of structural test points:

Algorithm `PC-to-TPs`$(P, \varphi, x)$.
Input: a PC decomposition $P$ of $\varphi$ w.r.t. $x$ containing exclusively PCs with precomputed bound choices $B$.
Output: a set $E$ of structural test points.

1. If there exists no candidate solution $(f, S, \tau)$ of a PC in $P$ with tag $\tau$ equal to "IP," "WLB," or "SLB," then return $\{(-\infty, (), \emptyset)\}$.

2. If there exists no candidate solution $(f, S, \tau)$ of a PC in $P$ with tag $\tau$ equal to "IP," "WUB," or "SUB," then return $\{(\infty, (), \emptyset)\}$.

3. $E := \emptyset$

4. For each prime constituent $(\pi, \Upsilon, \mathfrak{c}, \mathcal{F}, B) \in P$ do

   4.1. For each candidate solution $(f, S, \tau) \in \mathfrak{c}$ do

      4.1.1. If $l \in B$ and $\tau \in \{\text{IP}, \text{WLB}\}$, then add $\big((f, S), \pi, \mathcal{F}\big)$ to $E$.

      4.1.2. If $l \in B$ and $\tau \in \{\text{EP}, \text{SLB}\}$, then add $\big((f, S) + \varepsilon, \pi, \mathcal{F}\big)$ to $E$.

      4.1.3. If $u \in B$ and $\tau \in \{\text{IP}, \text{WUB}\}$, then add $\big((f, S), \pi, \mathcal{F}\big)$ to $E$.

      4.1.4. If $u \in B$ and $\tau \in \{\text{EP}, \text{SUB}\}$, then add $\big((f, S) - \varepsilon, \pi, \mathcal{F}\big)$ to $E$.

5. If there exists a prime constituent $(\pi, \Upsilon, \mathfrak{c}, \mathcal{F}, B) \in P$ such that $l \in B$, then add $(-\infty, (), \emptyset)$ to $E$.

6. If there exists a prime constituent $(\pi, \Upsilon, \mathfrak{c}, \mathcal{F}, B) \in P$ such that $u \in B$, then add $(\infty, (), \emptyset)$ to $E$.

7. Return $E$.

Algorithm `PC-to-TPs` is rather straightforward; as such it is an adjustment of step 5 of `vs-scheme`. Steps 1–2 detect whether the satisfying set $\Phi(\varphi, \mathbf{a})$ has no lower (upper) bound for any parameter values $\mathbf{a} \in \mathbb{R}^m$. In such special cases it is sufficient to substitute $\pm\infty$ into $\varphi$ to obtain an equivalent of $\exists x(\varphi)$, so an elimination set can be returned directly. Step 3 is reached if and only if there exist PCs $\mu_1$, $\mu_2 \in P$ such that $\mu_1$ has a candidate solutions of type from $\{\text{IP}, \text{WLB}, \text{SLB}\}$ and $\mu_2$ has a candidate solution of type from $\{\text{IP}, \text{WUB}, \text{SUB}\}$. In step 4 the candidate solutions are extracted from the prime constituents and structural test points are produced from them. For this the algorithm uses the bound choice $B \subseteq \{l, u\}$ of each prime constituent. The bound choice $B$ denotes whether lower bound or upper bound (or both) parametric root descriptions should be taken for a prime constituent. Steps 5–6 then include $\pm\infty$ into $E$ if needed.

The bound choice $B$ of each prime constituent has to be precomputed *before* calling `PC-to-TPs`. A precomputation of bound selections thus represents a bound selection strategy. A novel bound selection strategy will be the topic of Section 3.4. Our aim there will be to make meaningful bound choices by analyzing the structure of $\varphi$ along with a PC decomposition $P$ of $\varphi$. Furthermore, we will show that the set of test points computed by `PC-to-TPs` is indeed a structural elimination set for $\varphi$. In the meantime we will simply set $B$ either to $\{l\}$ or to $\{u\}$ to denote whether lower or upper bounds should be taken.

### 3.3.1 Two Preparatory Lemmas

Before presenting the structural QE scheme, we prove here two lemmas that we will use in the correctness proof of the scheme.

**Lemma 58.** *Let $\varphi(\mathbf{u}, x)$ be an $\wedge$-$\vee$-combination of atomic formulas. Let $\pi$ be a position in $\varphi$ such that $\pi(\varphi)$ is a Gauss formula with a set of candidate solutions $\mathfrak{c}$. Then $\varphi_\pi$ is a Gauss formula, and $\mathfrak{c}$ is a set of candidate solutions for $\varphi_\pi$.*

*Proof.* To begin with, observe that $\varphi_\pi \longleftrightarrow \pi(\varphi) \wedge \varphi'$ for some quantifier-free Tarski formula $\varphi'$. The reason for this is that we replaced all atomic formulas in $\varphi$ at positions independent from and not conjunctively associated with $\pi$ in $\varphi$ with "false." This replacement can be structurally seen as follows: Each $\vee$-node on the path from the root of $\varphi$ to position $\pi$ in $\varphi$ is replaced by its child lying on this path. In this way $\varphi_\pi$ consists either of $\pi(\varphi)$ (if there is no $\wedge$-node on the path) or contains $\pi(\varphi)$ at some position, and the path from the root of $\varphi_\pi$ to this position contains only $\wedge$-nodes.

Let now $\mathbf{a} \in \mathbb{R}^m$ be arbitrary parameter values. The equivalence we have just proven ensures that the following implication holds for any real number $\xi$: If $\xi \in \Phi\big(\Gamma(\varphi, \pi), \mathbf{a}\big)$, then $\xi \in \Phi(\pi(\varphi), \mathbf{a})$. Since $\Phi(\pi(\varphi), \mathbf{a})$ is finite and $\mathfrak{c}$ is a set of candidate solutions for $\pi(\varphi)$, we obtain that each boundary point of $\Phi\big(\Gamma(\varphi, \pi), \mathbf{a}\big)$ is properly covered by $\mathfrak{c}$. This shows that $\mathfrak{c}$ is a set of candidate solutions for $\Gamma(\varphi, \pi)$, which finishes the proof of the lemma. □

Notice that Lemma 58 does *not* hold for co-Gauss or atomic subformulas: Consider formula $x-1 \neq 0 \wedge x-a \geq 0$. Its subformula at position (1) is obviously co-Gauss. At the same time, $\varphi_{(1)} = \varphi$, but $\varphi$ is not co-Gauss because for any parameter values $\mathbf{a}$, there exist infinitely many real numbers that do not satisfy $\varphi(\mathbf{a})$—namely those that are strictly smaller than $a\langle\mathbf{a}\rangle$—i.e., the satisfying set of the formula is not co-finite.

**Lemma 59.** *Let $\varphi$ be a Gauss formula. Let $\mathfrak{c}$ be a set of candidate solutions for $\varphi$ containing only "IP" candidate solutions. Let $\mathbf{a} \in \mathbb{R}^m$ be parameter values. Define $\mathcal{E} = \big\{(f, S)\langle\mathbf{a}\rangle \in \mathbb{R} \mid (f, S, IP) \in \mathfrak{c}\big\}$. Assume that $\Phi(\varphi, \mathbf{a}) \neq \emptyset$. Then there exists $\xi \in \mathcal{E}$ such that $\mathbb{R} \models \varphi(\mathbf{a}, \xi)$.*

*Proof.* Since $\mathfrak{c}$ is a candidate solution set consisting only of "IP" candidate solutions, Theorem 19 guarantees that there exists $\xi \in \mathcal{E} \cup \{-\infty\}$ such that $\mathbb{R}^* \models \varphi(\mathbf{a}, \xi)$. If $\xi = -\infty$, then Lemma 17 implies that the satisfying set $\Phi(\varphi, \mathbf{a})$ is unbounded from below. On the other hand, $\varphi$ is a Gauss formula, so the satisfying set $\Phi(\varphi, \mathbf{a})$ is finite; a contradiction. Thus, $\xi \in \mathcal{E}$, and the lemma follows. □

### 3.3.2 The Scheme

Now we are finally ready to present a quantifier elimination scheme using structural virtual substitution. As usual we consider the elimination of $\exists x$ from $\exists x\big(\varphi(\mathbf{u}, x)\big)$, where $\varphi$ is an $\wedge$-$\vee$-combination of atomic formulas in the Tarski language $\mathfrak{L}$.

Algorithm `svs-scheme`$(\varphi, x)$.
Input: a quantifier-free Tarski formula $\varphi(\mathbf{u}, x)$, which is an $\wedge$-$\vee$-combination of atomic formulas, a variable $x$.
Output: a quantifier-free Tarski formula $\psi(\mathbf{u})$ equivalent to $\exists x(\varphi)$ or FAILED.

1. $P :=$ `PC-decomposition`$(\varphi, x)$
2. If $P$ is FAILED, then return FAILED.

3. For each prime constituent $(\pi, \Upsilon, \mathfrak{c}, \mathcal{F}, B) \in P$ do

   3.1. Set bound choice $B$ of $(\pi, \Upsilon, \mathfrak{c}, \mathcal{F}, B)$ to $\{l\}$.

4. $E := \texttt{PC-to-TPs}(P, \varphi, x)$

5. $\psi :=$ false

6. For each structural test point $(e, \pi, \mathcal{F}) \in E$ do

   6.1. If $e$ is $(f, S)$, then

      6.1.1. $\gamma_e := \texttt{guard}\big((f, S), x\big)$

      6.1.2. Compute $\varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} (f, S)]$ by replacing each atom $g \varrho 0$ occurring in $\varphi_\pi^{\mathcal{F}}$ with quantifier-free formula $\texttt{vs-at}(g \varrho 0, (f, S), x)$.

      6.1.3. $\psi := \psi \vee \big(\gamma_e \wedge \varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} (f, S)]\big)$

   6.2. If $e$ is $(f, S) + \varepsilon$, then

      6.2.1. $\gamma_e := \texttt{guard}\big((f, S), x\big)$

      6.2.2. Compute $\varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} (f, S) + \varepsilon]$ by replacing each atom $g \varrho 0$ occurring in $\varphi_\pi^{\mathcal{F}}$ with quantifier-free formula $\texttt{vs-at}\big(g \varrho 0, (f, S)+\varepsilon, x\big)$.

      6.2.3. $\psi := \psi \vee \big(\gamma_e \wedge \varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} (f, S) + \varepsilon]\big)$

   6.3. If $e$ is $\pm\infty$, then

      6.3.1. Compute $\varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} \pm\infty]$ by replacing each atom $g \varrho 0$ occurring in $\varphi_\pi^{\mathcal{F}}$ with quantifier-free formula $\texttt{vs-at}(g \varrho 0, \pm\infty, x)$.

      6.3.2. $\psi := \psi \vee \varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} \pm\infty]$

7. Return $\psi$.

Algorithm `svs-scheme` first computes a prime constituent decomposition in step 1. If this is successful, then it continues with step 3, which denotes by setting $B$ to $\{l\}$ that candidate solutions representing lower bounds should be taken. The call to `PC-to-TPs` in step 4 extracts from $P$ structural test points, adjusting them by $+\varepsilon$ when needed. Step 6 of the algorithm then substitutes the structural test points in $E$ into $\varphi$ by means of *structural* virtual substitution that takes advantage of the positional origin of a structural test point and the Boolean structure of $\varphi$.

Observe that $e$ in step 6 is usually of the form $(f, S)$, $(f, S) + \varepsilon$, or $-\infty$, because we take candidate solutions representing lower bounds. Nevertheless, it is possible that $E$ obtained in step 4 is $\big\{(\infty, (), \emptyset)\big\}$. This happens when $P$ does not contain prime constituents with "IP," "WUB," or "SUB" candidate solutions, i.e., the satisfying set $\Phi(\varphi, \mathbf{a})$ is guaranteed to be unbounded from above for any parameter values $\mathbf{a} \in \mathbb{R}^m$. For this reason step 6.3 handles $\infty$ as well.

Next we prove that this quantifier elimination algorithm scheme is correct:

**Theorem 60.** *Algorithm* `svs-scheme` *meets its specification.*

*Proof.* We proceed similarly as in the proof of Theorem 37, where we have proven that `vs-scheme` meets its specification. Let $\mathbf{a} \in \mathbb{R}^m$ be arbitrary parameter values. We prove that the following equivalence holds: $\mathbb{R} \models \psi(\mathbf{a})$ if and only if the satisfying set $\Phi(\varphi, \mathbf{a})$ is nonempty.

Assume first that $\mathbf{a}$ satisfies $\psi$. Observe that $\psi$ returned by `svs-scheme` is a quantifier-free disjunction obtained in step 6 by substituting each structural test point $(e, \pi, \mathcal{F}) \in E$ into $\varphi_\pi^{\mathcal{F}}$. Since $\mathbf{a}$ satisfies $\psi$, $\mathbf{a}$ has to satisfy at least one disjunction member of $\psi$. We distinguish three cases depending on the type of the structural test point that yielded this satisfied disjunction member:

1. $\mathbf{a}$ satisfies $\gamma \wedge \varphi_\pi^{\mathcal{F}}[x /\!\!/ (f, S)]$ for some $\big((f, S), \pi, \mathcal{F}\big) \in E$: Since `vs-at` meets its specification, and $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$, we use Theorem 25 to deduce that $\mathbb{R} \models \varphi_\pi^{\mathcal{F}}(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle)$. At the same time, $\varphi_\pi^{\mathcal{F}}$ was obtained from $\varphi$ by replacing some of its subformulas with "false." Since $\varphi$ is an $\wedge$-$\vee$-combination of atomic formulas, this implies that $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle)$, i.e., $\Phi(\varphi, \mathbf{a})$ is nonempty.

2. $\mathbf{a}$ satisfies $\gamma \wedge \varphi_\pi^{\mathcal{F}}[x /\!\!/ (f, S) + \varepsilon]$ for some $\big((f, S) + \varepsilon, \pi, \mathcal{F}\big) \in E$: Similarly to the previous case, since `vs-at` meets its specification, and $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$, we use Theorem 35 (i) to deduce that there exists a positive $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi_\pi^{\mathcal{F}}(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$. Again, using the fact that $\varphi_\pi^{\mathcal{F}}$ was derived from $\varphi$ by replacing some of its subformulas with "false" we obtain that $\mathbb{R} \models \varphi(\mathbf{a}, (f, S)\langle\mathbf{a}\rangle + \eta')$ for any positive $\eta' \in \mathbb{R}$ smaller than $\eta$. Thus, $\Phi(\varphi, \mathbf{a})$ is obviously nonempty.

3. $\mathbf{a}$ satisfies $\varphi_\pi^{\mathcal{F}}[x /\!\!/ \pm\infty]$: Since `vs-at` meets its specification, we use Theorem 35 (ii) to deduce that there exists $\eta \in \mathbb{R}$ such that $\mathbb{R} \models \varphi_\pi^{\mathcal{F}}(\mathbf{a}, \eta')$ for any $\eta' \in \mathbb{R}$ greater/smaller than $\eta$. Observe that `PC-to-TPs` always produces only structural test points of the form $(\pm\infty, (), \emptyset)$, so $\varphi_\pi^{\mathcal{F}}$ is in this case identical with $\varphi$. Therefore, $\Phi(\varphi, \mathbf{a})$ is unbounded from above/below and in particular nonempty.

Now we prove the converse implication. Assume that the satisfying set $\Phi(\varphi, \mathbf{a})$ is nonempty. We prove that $\mathbb{R} \models \psi(\mathbf{a})$, i.e., $\mathbf{a}$ satisfies at least one disjunct constructed in step 6 of `svs-scheme`.

If $E$ obtained in step 4 of `svs-scheme` is $\big\{(\infty, (), \emptyset)\big\}$, then $P$ does not contain prime constituents with "IP," "WUB," or "SUB" candidate solutions. This means that the satisfying set $\Phi(\varphi, \mathbf{a})$ is guaranteed to be unbounded from above for any parameter values $\mathbf{a} \in \mathbb{R}^m$, so $\mathbf{a}$ satisfies $\varphi[x /\!\!/ \infty]$, which is the only disjunct of $\psi$ in this case. In the following we assume that this special case does not occur, i.e., $P$ is nonempty and $E \neq \big\{(\infty, (), \emptyset)\big\}$.

Similarly, if $E$ obtained in step 4 of `svs-scheme` is $\big\{(-\infty, (), \emptyset)\big\}$, then $P$ does not contain prime constituents with "IP," "WLB," or "SLB" candidate solutions, and we deduce that $\mathbf{a}$ satisfies $\varphi[x /\!\!/ -\infty]$. In the following we therefore assume that there exist PCs $\mu_1$, $\mu_2 \in P$ with candidate solutions of types from $\{\mathrm{IP}, \mathrm{WUB}, \mathrm{SUB}\}$ and from $\{\mathrm{IP}, \mathrm{WLB}, \mathrm{SLB}\}$, respectively.

Let $\pi_1, \ldots, \pi_n$ be the positions of Gauss prime constituents in the order as found by `PC-decomposition` (cf. line 8 of that algorithm). Using Theorem 56 incrementally $n$-times—as was demonstrated in Example 57—for these $n$ Gauss positions we obtain:

$$\varphi \longleftrightarrow \varphi_{\pi_1} \vee \varphi_{\pi_2}^{\{\pi_1\}} \vee \cdots \vee \varphi_{\pi_n}^{\{\pi_1, \ldots, \pi_{n-1}\}} \vee \varphi^{\{\pi_1, \ldots, \pi_n\}}.$$

Since we assume $\Phi(\varphi, \mathbf{a}) \neq \emptyset$, we obtain that the satisfying set of at least one of the $n+1$ disjuncts on the right hand side of the equivalence above is nonempty.

Let $i$ be an index, counting from the left to the right, with this property. We distinguish two cases.

In the first case we have $i \in \{1, \ldots, n\}$. This means that the satisfying set $\Phi\big(\varphi_{\pi_i}^{\{\pi_1,\ldots,\pi_{i-1}\}}, \mathbf{a}\big)$ is nonempty. Lemma 58 ensures the following: First, $\varphi_{\pi_i}^{\{\pi_1,\ldots,\pi_{i-1}\}}$ is a Gauss formula. Second, a candidate solution set for $\pi_i(\varphi)$ is also a candidate solution set for $\varphi_{\pi_i}^{\{\pi_1,\ldots,\pi_{i-1}\}}$. Since the formula $\pi_i(\varphi)$ was found by `PC-decomposition` as a Gauss formula, its candidate solution set consists exclusively of IP candidate solutions. Lemma 59 therefore ensures that there exists $\xi \in \mathbb{R}$ such that $\mathbb{R} \models \varphi_{\pi_i}^{\{\pi_1,\ldots,\pi_{i-1}\}}(\mathbf{a}, \xi)$, where $\xi = (f, S)\langle \mathbf{a} \rangle$ for a candidate solution $(f, S, \mathrm{IP})$, which is in the candidate solution set for $\pi_i(\varphi)$. Since $\xi = (f, S)\langle \mathbf{a} \rangle$, $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$. Theorem 25 then ensures that $\mathbb{R} \models \gamma \wedge \big(\varphi_{\pi_i}^{\{\pi_1,\ldots,\pi_{i-1}\}}\big)[x \mathbin{/\!\!/} (f, S)](\mathbf{a})$. Algorithms `PC-decomposition` and `PC-to-TPs` now ensure that $\big((f, S), \pi_i, \{\pi_1, \ldots, \pi_{i-1}\}\big) \in E$. Thus, $\mathbf{a}$ satisfies the disjunct $\gamma \wedge \big(\varphi_{\pi_i}^{\{\pi_1,\ldots,\pi_{i-1}\}}\big)[x \mathbin{/\!\!/} (f, S)]$ constructed during an iteration of the loop in step 6 of `svs-scheme`.

In the second case we have $i = n+1$, so $\Phi(\varphi^{\{\pi_1,\ldots,\pi_n\}}, \mathbf{a})$ is nonempty. Since $\varphi^{\{\pi_1,\ldots,\pi_n\}}$ is an $\wedge$-$\vee$-combination of atomic formulas, the union of the candidate solution sets of all prime constituents in $P$ with type COGAUSS or AT yields, by Proposition 15 and induction, a candidate solution set for $\varphi^{\{\pi_1,\ldots,\pi_n\}}$. We distinguish three cases:

1. The satisfying set $\Phi(\varphi^{\{\pi_1,\ldots,\pi_n\}}, \mathbf{a})$ is bounded from below, and contains its infimum $\beta$. Since $\varphi^{\{\pi_1,\ldots,\pi_n\}}$ is an $\wedge$-$\vee$-combination of atomic formulas, there exists a prime constituent at position $\pi$ such that $\beta$ is an isolated point or a weak lower bound of $\Phi(\pi(\varphi), \mathbf{a})$. Therefore, there exists a candidate solution $(f, S, \tau)$ of the formula $\pi(\varphi)$ such that $\beta = (f, S)\langle \mathbf{a} \rangle$ and $\tau$ is equal to "IP" or "WLB." Algorithms `PC-decomposition` and `PC-to-TPs` ensure that $\big((f, S), \pi, \{\pi_1, \ldots, \pi_n\}\big) \in E$. Since $\beta = (f, S)\langle \mathbf{a} \rangle$, $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$. Theorem 25 then ensures that $\mathbf{a}$ satisfies $\gamma \wedge \varphi^{\{\pi_1,\ldots,\pi_n\}}[x \mathbin{/\!\!/} (f, S)]$.

   Now we prove that we also have $\mathbb{R} \models \gamma \wedge \big(\varphi_{\pi}^{\{\pi_1,\ldots,\pi_n\}}\big)[x \mathbin{/\!\!/} (f, S)](\mathbf{a})$. Observe that $(\mathbf{a}, \beta)$ satisfy both $\varphi^{\{\pi_1,\ldots,\pi_n\}}$ *and* its subformula $\pi(\varphi) = \pi(\varphi^{\{\pi_1,\ldots,\pi_n\}})$. Therefore, $(\mathbf{a}, \beta)$ satisfy the formula $\varphi_{\pi}^{\{\pi_1,\ldots,\pi_n\}}$ as well: The reason is that each subformula rooted at an $\vee$-node lying on the path from the root of $\varphi$ to the node at position $\pi$ is satisfied by $(\mathbf{a}, \beta)$ whenever $\pi(\varphi)$ is satisfied by $(\mathbf{a}, \beta)$. Consequently, replacing each atomic formula of $\varphi^{\{\pi_1,\ldots,\pi_n\}}$ at a position independent from and not conjunctively associated with $\pi$ with "false" does not make the formula unsatisfied at $(\mathbf{a}, \beta)$. Thus, $(\mathbf{a}, \beta)$ satisfy $\varphi_{\pi}^{\{\pi_1,\ldots,\pi_n\}}$. Recall that $\beta = (f, S)\langle \mathbf{a} \rangle$, so Theorem 25 ensures that $\mathbb{R} \models \gamma \wedge \big(\varphi_{\pi}^{\{\pi_1,\ldots,\pi_n\}}\big)[x \mathbin{/\!\!/} (f, S)](\mathbf{a})$, where $\gamma$ is a guard of $(f, S)$.

2. The satisfying set $\Phi(\varphi^{\{\pi_1,\ldots,\pi_n\}}, \mathbf{a})$ is bounded from below, but it does not contain its infimum $\beta$. We proceed similarly as in the previous case: Since $\varphi^{\{\pi_1,\ldots,\pi_n\}}$ is an $\wedge$-$\vee$-combination of atomic formulas, there exists a prime constituent at position $\pi$ such that $\beta$ is an excluded point or a strict lower bound of $\Phi(\pi(\varphi), \mathbf{a})$. Therefore, there exists a candidate solution $(f, S, \tau)$ of $\pi(\varphi)$ such that $\beta = (f, S)\langle \mathbf{a} \rangle$ and $\tau$ is equal

to "EP" or "SLB." Algorithms `PC-decomposition` and `PC-to-TPs` ensure that $\big((f,S)+\varepsilon,\pi,\{\pi_1,\ldots,\pi_n\}\big) \in E$. Since $\beta = (f,S)\langle\mathbf{a}\rangle$, $\mathbf{a}$ satisfies a guard $\gamma$ of $(f,S)$. Theorem 35 (i) then ensures that $\mathbf{a}$ satisfies $\gamma \wedge \varphi^{\{\pi_1,\ldots,\pi_n\}}[x \mathbin{/\!\!/} (f,S)+\varepsilon]$.

To prove that $\mathbf{a}$ also satisfies $\gamma \wedge \big(\varphi_\pi^{\{\pi_1,\ldots,\pi_n\}}\big)[x \mathbin{/\!\!/} (f,S)+\varepsilon]$ one proceeds as follows: First observe that our assumptions guarantee that there exists a positive $\eta \in \mathbb{R}$ such that for each positive $\eta' \in \mathbb{R}$ smaller than $\eta$ we have $\beta+\eta' \in \big(\Phi(\varphi^{\{\pi_1,\ldots,\pi_n\}},\mathbf{a}) \cap \Phi(\pi(\varphi),\mathbf{a})\big)$. Then use the same structural argument as in the previous case to deduce that the replacement of each atomic formula of $\varphi^{\{\pi_1,\ldots,\pi_n\}}$ at a position independent from and not conjunctively associated with $\pi$ with "false" does not make the formula unsatisfied at $(\mathbf{a},\beta+\eta')$ for any positive $\eta'$ smaller than $\eta$. Using Theorem 35 (i) we then obtain that $\mathbf{a}$ satisfies $\gamma \wedge \big(\varphi_\pi^{\{\pi_1,\ldots,\pi_n\}}\big)[x \mathbin{/\!\!/} (f,S)+\varepsilon]$.

3. The satisfying set $\Phi(\varphi^{\{\pi_1,\ldots,\pi_n\}},\mathbf{a})$ is unbounded from below. In this case we use Theorem 35 (ii) to deduce $\mathbb{R} \models \varphi^{\{\pi_1,\ldots,\pi_n\}}[x \mathbin{/\!\!/} -\infty](\mathbf{a})$, which implies $\mathbb{R} \models \varphi[x \mathbin{/\!\!/} -\infty](\mathbf{a})$. Since we set $B$ of all prime constituents in $P$ to $\{l\}$, `PC-to-TPs` includes $(-\infty,(),\emptyset)$ in $E$. Therefore, $\varphi[x \mathbin{/\!\!/} -\infty]$ is indeed a member of the disjunction $\psi$ returned by `svs-scheme`.

In all there cases we have just proven that there exists a disjunct of $\psi$ returned by `svs-scheme` that is satisfied by $\mathbf{a}$ whenever the satisfying set $\Phi(\varphi,\mathbf{a})$ is nonempty. This finishes the proof of the theorem. □

Observe that Algorithm `svs-scheme` includes in $E$ only test points that possibly represent a lower bound of a satisfying set. Similarly as with `vs-scheme`, it is straightforward to change the scheme and the proof to argue that upper bound test points can be used instead. Furthermore, it is also correct to decide between these two options after an analysis of the PC decomposition $P$. Yet another bound selection strategy looking at the Boolean structure of $\varphi$ will be the subject of the next section.

At this point let us remind ourselves that with `vs-scheme` we were also able to construct a set of test points for $\varphi$ and $x$ à la Theorem 19 (ii). This meant that we were able to use WLB candidate solutions instead of EP candidate solutions when constructing an elimination set for $\varphi$ and $x$. A careful analysis of the proof of Theorem 60 reveals that we can proceed similarly in this case as well. Even though we do not use Theorem 19 directly for the "non-Gauss" part of the formula $\varphi$, the correctness proof can be adjusted so that EP candidate solutions are exchanged for WLB candidate solutions when needed; of course, after a corresponding straightforward adjustment of `PC-to-TPs` that adds a structural test point of the form $\big((f,S)+\varepsilon,\pi,\mathcal{F}\big)$ to $E$ for WLB candidate solutions instead of EP candidate solutions. Similarly as with `vs-scheme` of Chapter 2, this strategy avoiding excluded points is beneficial whenever there are more excluded points than lower bound candidate solutions. As a special case, the correctness of this strategy ensures that both $\big\{(-\infty,(),\emptyset)\big\}$ and $\big\{(\infty,(),\emptyset)\big\}$ are elimination sets for any formula whose PC decomposition consists of exclusively co-Gauss prime constituents.

Before we delve into the topic of our bound selection strategy, we first discuss one principle difference between `vs-scheme` and `svs-scheme`.

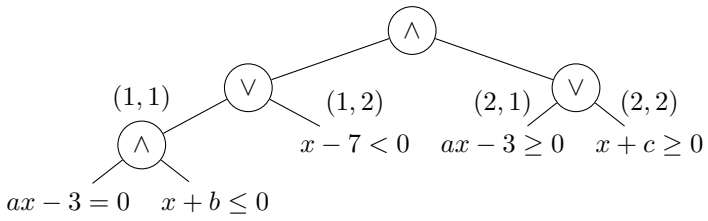



Figure 3.8: Formula $\varphi$ from Example 61.

### 3.3.3 Conflation

In contrast to non-structural elimination sets, it is possible in the structural context here that one parametric root description occurs in more than one structural test point. We illustrate such a situation with an example.

**Example 61.** Consider the quantifier-free formula $\varphi$ in Figure 3.8. Algorithm `PC-decomposition` discovers four prime constituents, namely one Gauss prime constituent at position $(1,1)$ and three atomic prime constituents at positions $(1,2)$, $(2,1)$, and $(2,2)$. Assume for simplicity that we do not use clustering and that lower bound candidate solutions are taken. Therefore, we obtain from `PC-to-TPs` a set $E$ consisting of the following five structural test points for $\varphi$. For clarity we annotated parametric root description, position, and false replacement set in each structural test point:

$$\Big(\underbrace{\big(ax-3,(1,1)\big)}_{e},\underbrace{(1,1)}_{\pi},\underbrace{\emptyset}_{\mathcal{F}}\Big),\ \Big(\underbrace{\big(ax-3,(-1,1)\big)}_{e},\underbrace{(1,1)}_{\pi},\underbrace{\emptyset}_{\mathcal{F}}\Big),$$
$$\Big(\underbrace{\big(ax-3,(1,1)\big)}_{e},\underbrace{(2,1)}_{\pi},\underbrace{\{(1,1)\}}_{\mathcal{F}}\Big),\ \Big(\underbrace{\big(x+c,(1,1)\big)}_{e},\underbrace{(2,2)}_{\pi},\underbrace{\{(1,1)\}}_{\mathcal{F}}\Big),$$
$$\Big(\underbrace{-\infty}_{e},\underbrace{()}_{\pi},\underbrace{\emptyset}_{\mathcal{F}}\Big).$$

As we can see, the parametric root description $e = \big(ax-3,(1,1)\big)$ occurs here twice with two distinct positions and false replacement sets. ◇

When $(e,\pi_1,\mathcal{F}_1)$ and $(e,\pi_2,\mathcal{F}_2)$ occur in $E$ we have to substitute $e=(f,S)$ (or $e=(f,S)\pm\varepsilon$) into two different formulas, namely $\varphi_{\pi_1}^{\mathcal{F}_1}$ and $\varphi_{\pi_2}^{\mathcal{F}_2}$. The idea of conflation is to prevent this kind of redundancy, and substitute $e$ into one formula $\varphi_{\pi'}^{\mathcal{F}'}$ instead. The position $\pi'$ and the false replacement set $\mathcal{F}'$ are computed from $\pi_1$, $\mathcal{F}_1$ and $\pi_2$, $\mathcal{F}_2$ by "conflating" them together: $\pi'$ is the lowest common ancestor of $\pi_1$ and $\pi_2$, and $\mathcal{F}'$ is the intersection of $\mathcal{F}_1$ and $\mathcal{F}_2$. This is made explicit in the following algorithm:

Algorithm `TPs-conflate`$(E)$.
Input: a set $E$ of structural test points.
Output: a set $E'$ of structural test points, which is obtained from $E$ by "conflating everything what can be conflated," i.e., $E'$ contains no duplicates of non-structural test points of the forms $(f,S)$, $(f,S)+\varepsilon$, or $(f,S)-\varepsilon$.

1. $E' := \emptyset$

2. While $E \neq \emptyset$ do

   2.1. Pop $t_1 = \big((f_1, S_1) + \iota_1, \pi_1, \mathcal{F}_1\big)$ from $E$.

   2.2. $T := E$

   2.3. While $T \neq \emptyset$ do

      2.3.1. Pop $t_2 = \big((f_2, S_2) + \iota_2, \pi_2, \mathcal{F}_2\big)$ from $T$.

      2.3.2. If $f_1 = f_2$ and $\iota_1 = \iota_2$ and $S_1 \subseteq S_2$, then

         2.3.2.1. $t_1 := \big((f_2, S_2) + \iota_2, \mathrm{lca}(\pi_1, \pi_2), \mathcal{F}_1 \cap \mathcal{F}_2\big)$

         2.3.2.2. Delete $t_2$ from $E$.

      2.3.3. If $f_1 = f_2$ and $\iota_1 = \iota_2$ and $S_2 \subseteq S_1$, then

         2.3.3.1. $t_1 := \big((f_1, S_1) + \iota_1, \mathrm{lca}(\pi_1, \pi_2), \mathcal{F}_1 \cap \mathcal{F}_2\big)$

         2.3.3.2. Delete $t_2$ from $E$.

   2.4. $E' := E' \cup \{t_1\}$

3. Return $E'$.

Observe that `TPs-conflate` meets its specification and runs in time $O\big(|E|^2\big)$. Note that the nonstandard parts $\iota_1$, $\iota_2$ in the loop in step 2.3 can be 0, $\varepsilon$, or $-\varepsilon$. Since we will call `TPs-conflate` only on $E$ obtained from `PC-to-TPs` for a set of prime constituents computed by `PC-decomposition`, we will *always* have either $\mathcal{F}_1 \subseteq \mathcal{F}_2$ or $\mathcal{F}_2 \subseteq \mathcal{F}_1$ in the loop in step 2.3.

For the set $E$ from Example 61 we obtain `TPs-conflate`$(E)$:

$$\Big(\big(ax - 3, (1,1)\big), (), \emptyset\Big), \Big(\big(ax - 3, (-1,1)\big), (1,1), \emptyset\Big),$$
$$\Big(\big(x + c, (1,1)\big), (2,2), \{(1,1)\}\Big), \big(-\infty, (), \emptyset\big).$$

Observe that without conflation `svs-scheme` ensures that $\{\pi\} \cup \mathcal{F}$ is a set of pairwise independent positions for any test point of the form $(e, \pi, \mathcal{F})$ returned by `PC-to-TPs`. Using conflation this is not guaranteed anymore, because we use the lca operator that possibly causes that a position $\mathrm{lca}(\pi_1, \pi_2)$ is dependent with a position from $\mathcal{F}_1 \cap \mathcal{F}_2$. At the same time this is not a formal problem at all, because the deletion and condensing operators do not change the Boolean structure of a formula.

The use of `TPs-conflate` in the context of `svs-scheme` is straightforward: Call it to conflate the set of test points in `svs-scheme` before substituting them into $\varphi$. The correctness of this approach is asserted by the following theorem:

**Theorem 62** (Structural Quantifier Elimination Scheme with Conflation)**.** *Replace line 4 of* `svs-scheme` *with*

$$E := \texttt{TPs-conflate}\big(\texttt{PC-to-TPs}(P, \varphi, x)\big).$$

*This modified scheme called "*`svs-scheme` *with condensing" is correct.*

*Proof.* Let $\psi$ be the formula returned by `svs-scheme` with condensing. Let $\mathbf{a} \in \mathbb{R}^m$ be arbitrary parameter values. We prove that the following equivalence holds: $\mathbb{R} \models \psi(\mathbf{a})$ if and only if the satisfying set $\Phi(\varphi, \mathbf{a})$ is nonempty.

Assume first that $\mathbf{a}$ satisfies $\psi$. The proof of the fact that $\Phi(\varphi, \mathbf{a}) \neq \emptyset$ is literally the same as in the proof of Theorem 60, and therefore we omit it here.

Now assume that the satisfying set $\Phi(\varphi, \mathbf{a})$ is nonempty. In the following we prove that $\mathbb{R} \models \psi(\mathbf{a})$, i.e., $\mathbf{a}$ satisfies at least one disjunct of $\psi$ returned by `svs-scheme` with condensing. We proceed similarly as in the proof of Theorem 60. There we have proven that there exists $(e, \pi, \mathcal{F}) \in E$ such that $\mathbb{R} \models \gamma \wedge \varphi_\pi^{\mathcal{F}}[x /\!\!/ e](\mathbf{a})$. We emphasize that $E$ here is the set of test points computed by `svs-scheme`. In the following we denote the set computed by `svs-scheme` with condensing by $E'$.

If $e = \pm\infty$, then observe that `TPs-conflate` does not modify structural test points of the form $(\pm\infty, (), \emptyset)$, i.e., $(\pm\infty, (), \emptyset) \in E'$. Therefore, $\varphi[x /\!\!/ \pm\infty]$ is a disjunct constructed during the loop in step 6 of `svs-scheme` with condensing. Since we have proven that $\mathbb{R} \models \gamma \wedge \varphi_\pi^{\mathcal{F}}[x /\!\!/ e](\mathbf{a})$ in this case, it follows that $\mathbf{a}$ satisfies a disjunct of $\psi$ returned by `svs-scheme` with condensing.

Assume now that $e$ is of the form $(f, S) + \iota$, where $\iota$ is 0 or $\varepsilon$. There are two cases to consider:

1. $(e, \pi, \mathcal{F}) \in E'$: This means that $(e, \pi, \mathcal{F})$ was left intact during the loop in step 2 of `TPs-conflate`. In this case, again, $\gamma \wedge \varphi_\pi^{\mathcal{F}}$ was obviously constructed by `svs-scheme` with condensing, and $\mathbb{R} \models \gamma \wedge \varphi_\pi^{\mathcal{F}}[x /\!\!/ e](\mathbf{a})$ holds. Thus, $\mathbf{a}$ satisfies $\psi$ returned by `svs-scheme` with condensing.

2. $(e, \pi, \mathcal{F}) \notin E'$: This means that $(e, \pi, \mathcal{F})$ was replaced by another structural test point during the loop in step 2 of `TPs-conflate`. The replacement of structural test points in `TPs-conflate` is done in such a way that there exists another structural test point $(e', \pi', \mathcal{F}') \in E'$ such that $e' = (f, S') + \iota$, $S \subseteq S'$, $\pi \sqsubseteq \pi'$, and $\mathcal{F}' \subseteq \mathcal{F}$. By the construction, neither $S$ nor $S'$ contain any duplicate root specifications, so $(f, S')\langle\mathbf{a}\rangle$ is well-defined, and $(f, S')\langle\mathbf{a}\rangle = (f, S)\langle\mathbf{a}\rangle$. Furthermore, any guard of $(f, S)$ implies any guard of $(f, S')$. At the same time, Proposition 53 together with $\pi \sqsubseteq \pi'$ and $\mathcal{F}' \subseteq \mathcal{F}$ ensures that $\varphi_\pi^{\mathcal{F}}$ implies $\varphi_{\pi'}^{\mathcal{F}'}$.

   Putting this all together we obtain: There exists $\left(e', \pi', \mathcal{F}'\right) \in E'$ such that $\mathbb{R} \models \gamma \wedge \varphi_\pi^{\mathcal{F}}[x /\!\!/ e](\mathbf{a})$ implies $\mathbb{R} \models \gamma' \wedge \varphi_{\pi'}^{\mathcal{F}'}[x /\!\!/ e'](\mathbf{a})$, where $\gamma'$ is a guard of $e'$. Since we have proven that $\mathbb{R} \models \gamma \wedge \varphi_\pi^{\mathcal{F}}[x /\!\!/ e](\mathbf{a})$, we obtain that $\mathbb{R} \models \gamma' \wedge \varphi_{\pi'}^{\mathcal{F}'}[x /\!\!/ e'](\mathbf{a})$.

   Finally, the formula $\gamma' \wedge \varphi_{\pi'}^{\mathcal{F}'}[x /\!\!/ e']$ was constructed during the loop in step 6 of `svs-scheme` with condensing, so we obtain that $\mathbf{a}$ satisfies at least one disjunct of the formula $\psi$ returned by the modified scheme. □

It is possible that other than the "conflate everything possible" strategy lead to reasonable results. Nevertheless, the aggressive conflation strategy of `TPs-conflate`, whose correctness we have just proven, plays a prominent role: Using it we can really state that `svs-scheme` is a generalization of `vs-scheme` in the following sense: Whenever $(e, \pi, \mathcal{F})$ is substituted into $\varphi$, then $e$ is substituted into $\varphi$ by `vs-scheme` as well. Furthermore, $e$ is not substituted into $\varphi$ twice (with different originating positions and false replacement sets) by `svs-scheme` with conflation.

We conclude this section with an example where "non-conflation" could make more sense than conflation: Consider a formula $\varphi$ of the form $\varphi_1 \vee \varphi_2 \vee \varphi_3$ and assume that structural test points $(e, \pi_1, \emptyset)$ and $(e, \pi_2, \emptyset)$, where $\pi_i$ lies in subformula $\varphi_i$, are computed by `svs-scheme`. While substituting the two structural test points into $\varphi$, the algorithm includes in the output disjunction the

two formulas $\varphi_{\pi_1}[x /\!\!/ e]$ and $\varphi_{\pi_2}[x /\!\!/ e]$. Using conflation the algorithm includes the formula $\varphi[x /\!\!/ e]$ in the output instead, because $\mathrm{lca}(\pi_1, \pi_2) = ()$. In the former case the number of atoms we need to substitute $e$ into is at most $|\varphi_1| + |\varphi_2|$; in the latter case the number of atoms we need to substitute $e$ into is at most $|\varphi_1| + |\varphi_2| + |\varphi_3|$, because the replacement with "false" during condensing of $\varphi$ w.r.t. $\pi_1$ and $\pi_2$ can be seen as simply deleting subformulas $\varphi_2$, $\varphi_3$ and $\varphi_1$, $\varphi_3$, respectively. Obviously, this can make a difference—in both running time and length of the computed quantifier-free equivalent—in cases when $\psi_3$ is a long formula. By $|\varphi_i|$ we denote here the number of atomic formulas in $\varphi_i$.

In contrast, it is not hard to come up with an example where the strategy "conflate everything possible" is more reasonable. It seems that one could use optimization techniques along with counting the numbers of target atomic formulas to decide whether and how $e$, occurring in a structural elimination set at least twice, should be conflated before substitution. A systematic investigation of this is left for future work.

## 3.4 Bound Selection

Our quantifier elimination schemes `vs-scheme` and `svs-scheme` globally decide to take either exclusively lower or exclusively upper bound candidate solutions to obtain a (structural) elimination set for $\varphi$ and $x$. In this section we show how to compute elimination sets, which are not obtained by such a global decision. We present a bound selection strategy based on an analysis of the conjunctive associativity relation on positions combined with 0-1 Integer Linear Programming (0-1 ILP). The approach possibly yields a heterogeneous structural elimination set, which contains "a mixture" of structural test points obtained from lower bound candidate solutions and upper bound candidate solutions.

As usual let $\varphi$ be an $\wedge$-$\vee$-combination of Tarski atomic formulas. Let $P$ be a prime constituent decomposition of $\varphi$. The *conjunctive associativity graph for $\varphi$ and $P$* denoted by $G(\varphi, P) = (V, E)$ is defined as follows:

$$\begin{aligned} V &= \big\{\pi \in \mathtt{Pos}(\varphi) \mid (\pi, \Upsilon, \mathfrak{c}, \mathcal{F}, B) \in P \text{ and } \Upsilon \in \{\mathrm{COGAUSS}, \mathrm{AT}\}\big\}, \\ E &= \big\{(\pi_i, \pi_j) \in V \times V \mid \pi_i \text{ and } \pi_j \text{ are conjunctively associated in } \varphi\big\}. \end{aligned}$$

An *admissible coloring* of $G(\varphi, P) = (V, E)$ is a mapping $\chi : V \to 2^{\{l,u\}}$ such that for every $\pi \in V$ we have $\chi(\pi) \neq \emptyset$, and for every $(\pi_i, \pi_j) \in E$ we have $\chi(\pi_i) \cap \chi(\pi_j) \neq \emptyset$. In the following we show that there is a one-to-one correspondence between all admissible colorings of $G(\varphi, P) = (V, E)$ and all solutions of the system

$$\bigwedge_{\pi \in V} \big(x^l_\pi + x^u_\pi \geq 1\big), \qquad \bigwedge_{(\pi_i, \pi_j) \in E} \big(x^l_{\pi_i} + x^u_{\pi_j} \geq 1 \wedge x^u_{\pi_i} + x^l_{\pi_j} \geq 1\big) \tag{3.1}$$

of 0-1 ILP constraints in variables $\bigcup_{\pi \in V} \{x^l_\pi, x^u_\pi\}$.

First we show that a solution of system (3.1) gives rise to an admissible coloring of the conjunctive associativity graph $G(\varphi, P)$.

**Proposition 63.** *Let $s$ be a 0-1 assignment to variables $\bigcup_{\pi\in V}\{x^l_\pi, x^u_\pi\}$ satisfying system (3.1). Define $\chi$ as follows:*

$$\chi(\pi) = \begin{cases} \emptyset & \text{if } s(x^l_\pi) = 0 \text{ and } s(x^u_\pi) = 0, \\ \{l\} & \text{if } s(x^l_\pi) = 1 \text{ and } s(x^u_\pi) = 0, \\ \{u\} & \text{if } s(x^l_\pi) = 0 \text{ and } s(x^u_\pi) = 1, \\ \{l, u\} & \text{if } s(x^l_\pi) = 1 \text{ and } s(x^u_\pi) = 1. \end{cases}$$

*Then $\chi$ is an admissible coloring of $G(\varphi, P) = (V, E)$.*

*Proof.* To begin with, observe that the condition $s(x^l_\pi) + s(x^u_\pi) \geq 1$ implies $\chi(\pi) \neq 0$ for every $\pi \in V$.

Let $(\pi_i, \pi_j) \in E$, and assume that $\chi(\pi_i) \cap \chi(\pi_j) = \emptyset$. Since $\chi(\pi) \neq \emptyset$ for all $\pi \in V$, we have

1. either $\chi(\pi_i) = \{u\}$ and $\chi(\pi_j) = \{l\}$, or
2. $\chi(\pi_i) = \{l\}$ and $\chi(\pi_j) = \{u\}$.

In the first case we obtain that $s(x^l_{\pi_i}) = s(x^u_{\pi_j}) = 0$, i.e., $s$ does not satisfy the constraint $x^l_{\pi_i} + x^u_{\pi_j} \geq 1$ of system (3.1); a contradiction. In the second case we obtain that $s(x^u_{\pi_i}) = 0$ and $s(x^l_{\pi_j}) = 0$, i.e., $s$ does not satisfy the constraint $x^u_{\pi_i} + x^l_{\pi_j} \geq 1$ of system (3.1), which is again a contradiction. This proves that $\chi$ is an admissible coloring of $G(\varphi, P)$. □

Next we show that an admissible coloring can be converted to a solution of system (3.1).

**Proposition 64.** *Let $\chi$ be an admissible coloring of $G(\varphi, P) = (V, E)$. Define*

$$s(x^l_\pi) = \begin{cases} 0 & \text{if } l \notin \chi(\pi) \\ 1 & \text{if } l \in \chi(\pi) \end{cases} \quad \text{and} \quad s(x^u_\pi) = \begin{cases} 0 & \text{if } u \notin \chi(\pi) \\ 1 & \text{if } u \in \chi(\pi). \end{cases}$$

*Then $s$ satisfies system (3.1) of 0-1 ILP constraints.*

*Proof.* To begin with, notice that for all $\pi \in V$ the condition $\chi(\pi) \neq \emptyset$ implies that $s$ satisfies the constraint $x^l_\pi + x^u_\pi \geq 1$ of system (3.1).

Let $(\pi_i, \pi_j) \in E$. Since $\chi$ is an admissible coloring of $G(\varphi, P)$, we have $\chi(\pi_i) \cap \chi(\pi_j) \neq \emptyset$. There are two cases to consider:

1. $l \in \chi(\pi_i) \cap \chi(\pi_j)$: The definition of $s$ ensures that $s(x^l_{\pi_i}) = s(x^l_{\pi_j}) = 1$, i.e., $s$ satisfies constraints $x^l_{\pi_i} + x^u_{\pi_j} \geq 1$ and $x^u_{\pi_i} + x^l_{\pi_j} \geq 1$ of system (3.1).
2. $u \in \chi(\pi_i) \cap \chi(\pi_j)$: The definition of $s$ ensures that $s(x^u_{\pi_i}) = s(x^u_{\pi_j}) = 1$, i.e., $s$ satisfies constraints $x^l_{\pi_i} + x^u_{\pi_j} \geq 1$ and $x^u_{\pi_i} + x^l_{\pi_j} \geq 1$ of system (3.1).

This shows that $s$ satisfies each and every constraint of system (3.1). □

From now on we will not really distinguish between a solution $s$ of system (3.1), an admissible coloring $\chi$ of $G(\varphi, P)$, and bound choices for $P$ induced by $\chi$. The following example illustrates a few admissible colorings of $G(\varphi, P)$, bound choices for $P$ induced by these colorings, and sets of structural elimination test points computed from $P$ by `PC-to-TPs` using these bound choices.

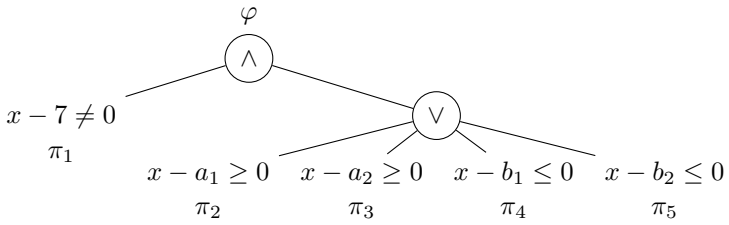



Figure 3.9: Formula $\varphi$ from Example 65 and its PC decomposition $P$ into five prime constituents at positions $\pi_1, \ldots, \pi_5$.

**Example 65.** Consider PC decomposition $P$ of $\varphi$ in Figure 3.9 into five prime constituents at positions $\pi_i$. The constituent at position $\pi_1$ is a co-Gauss prime constituent, and the other constituents are atomic prime constituents. The conjunctive associativity graph $G(\varphi, P)$ is shown in Figure 3.10.

We discuss three admissible colorings of $G(\varphi, P)$:

1. $\chi_l(\pi_i) = \{l\}$ for every $i \in \{1, \ldots, 5\}$: This coloring encodes the global decision to take lower bound candidate solutions. Using `PC-to-TPs`, this coloring gives rise to the following four structural test points:

$$\Big(\big(x - a_1, (1,1)\big), \pi_2, \emptyset\Big), \Big(\big(x - a_2, (1,1)\big), \pi_3, \emptyset\Big),$$
$$\Big(\big(x - 7, (1,1)\big) + \varepsilon, \pi_1, \emptyset\Big), \big(-\infty, (), \emptyset\big).$$

2. $\chi_u(\pi_i) = \{u\}$ for every $i \in \{1, \ldots, 5\}$: This coloring encodes the global decision to take upper bound candidate solutions. Using `PC-to-TPs` we obtain:

$$\Big(\big(x - b_1, (1,1)\big), \pi_4, \emptyset\Big), \Big(\big(x - b_2, (1,1)\big), \pi_5, \emptyset\Big),$$
$$\Big(\big(x - 7, (1,1)\big) - \varepsilon, \pi_1, \emptyset\Big), \big(\infty, (), \emptyset\big).$$

3. Consider the following admissible coloring $\chi$ of $G_\varphi$:

$$\chi(\pi_1) = \{l, u\}, \quad \chi(\pi_2) = \chi(\pi_3) = \{u\}, \quad \chi(\pi_4) = \chi(\pi_5) = \{l\}.$$

   Using `PC-to-TPs`, $\chi$ leads to the following four of structural test points:

$$\Big(\big(x - 7, (1,1)\big) - \varepsilon, \pi_1, \emptyset\Big), \Big(\big(x - 7, (1,1)\big) + \varepsilon, \pi_1, \emptyset\Big),$$
$$\big(-\infty, (), \emptyset\big), \big(\infty, (), \emptyset\big).$$

The results of the previous sections ensure that the first and the second set of structural test points are structural elimination sets for $\varphi$ and $x$. In the following we will show that the third set of structural test points is a structural elimination set for $\varphi$ and $x$ as well. ◇

Observe that each solution of system (3.1) ensures that bound choices of a pair of conjunctively associated prime constituents share at least one element

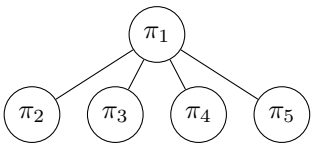



Figure 3.10: The conjunctive associativity graph for $\varphi$ and $P$ from Figure 3.9.

from $\{l, u\}$. In an admissible coloring, prime constituents that are not conjunctively associated do not need to share a bound choice, as we have seen in Example 65, where $\chi(\pi_2) \neq \chi(\pi_4)$ for an admissible coloring $\chi$.

The intuition behind this approach is best explained on the DNF of $\varphi$: Compute the DNF of $\varphi$ in our head, treating prime constituents as atomic formulas. Push the quantifier $\exists x$ in front of each disjunct, which is a conjunction of prime constituents. Then we are able to make a choice to take lower or upper bounds in each disjunct separately. Therefore, it suffices to ensure that only those prime constituents that are conjunctively associated in $\varphi$ share at least one element from $\{l, u\}$. The Gauss prime constituents are left out of this whole process, because they contain only IP candidate solutions, which represent both lower and upper bounds on $x$. Accordingly, `PC-to-TPs` produces the same structural test points for bound choices $\{l\}$, $\{u\}$, and $\{l, u\}$. For Gauss prime constituents we therefore only have to ensure that their bound choice $B$ is not $\emptyset$ before calling `PC-to-TPs`.

In the following we give a QE scheme based on this intuition and prove its correctness. Before doing so we first present an algorithm that for $\varphi$ and $P$ constructs system (3.1), solves it, and assigns bound choices to prime constituents in $P$ accordingly.

### 3.4.1 Procedure `PC-bs-ILP`

The following bound selection algorithm `PC-bs-ILP` is based on the idea that bound choices of two prime constituents need to share an element from $\{l, u\}$ if and only if they are conjunctively associated. To achieve this the algorithm constructs and solves the system (3.1) for any given PC decomposition $P$ of $\varphi$. As a result, the algorithm fixes the bound choice $B$ of each prime constituent in the PC decomposition $P$ of $\varphi$.

Steps 1–2 of the algorithm solve the 0-1 ILP system (3.1) for a PC decomposition $P$ of $\varphi$. For this we use `ILP-construct`, which first constructs the system (3.1) without explicitly constructing the conjunctive associativity graph $G(\varphi, P)$.

Steps 4–6 ensure that the bound choice $B$ of each Gauss prime constituent is nonempty, and that $B$ is not $\{l, u\}$. We ensure that the latter condition holds only to prevent including both $(-\infty, (), \emptyset)$ and $(\infty, (), \emptyset)$ into $E$ when $P$ contains exclusively Gauss prime constituents, because in such a case only one of $\pm\infty$ is needed.

Algorithm `PC-bs-ILP`$(P, \varphi, x)$.
Input: a PC decomposition $P$ of $\varphi$ w.r.t. $x$.
Output: a PC decomposition $P'$ of $\varphi$ w.r.t. $x$ containing only prime constituents with fixed bound choices $B$.

1. $(f, \mathcal{C}) := \texttt{ILP-construct}(P, \varphi, x)$
2. Compute a solution $s$ of the system $\mathcal{C}$.
3. For each $\mu \in P$ at position $\pi$ with type $\Upsilon \in \{\text{COGAUSS}, \text{AT}\}$ do
   3.1. $B := \emptyset$
   3.2. If $s(x_\pi^l) = 1$, then $B := B \cup \{l\}$.
   3.3. If $s(x_\pi^u) = 1$, then $B := B \cup \{u\}$.
   3.4. Fix the bound choice of $\mu$ to $B$.
4. $B_g := \{l\}$
5. If there exists a variable $x_\pi^u$ such that $s(x_\pi^u) = 1$, then $B_g := \{u\}$.
6. For each $\mu \in P$ with type $\Upsilon = \text{GAUSS}$ do
   6.1. Fix the bound choice of $\mu$ to $B_g$.
7. Return $P$.

Algorithm $\texttt{ILP-construct}(P, \varphi, x)$.
Input: a PC decomposition $P$ of $\varphi$ w.r.t. $x$.
Output: a system $\mathcal{C}$ of 0-1 ILP constraints for $\varphi$ and $P$ as in (3.1).

1. $(\mathcal{C}, P') := \texttt{ILP-subroutine}\big(P, \varphi, x, ()\big)$
2. For each $\mu = (\pi, \Upsilon, \mathfrak{c}, \mathcal{F}, B) \in P'$ do
   2.1. $\mathcal{C} := \mathcal{C} \cup \{x_\pi^l + x_\pi^u \geq 1\}$
3. Return $\mathcal{C}$.

Algorithm $\texttt{ILP-subroutine}(P, \varphi, x, \pi)$.
Input: a PC decomposition $P$ of $\varphi$ w.r.t. $x$, a position $\pi \in \texttt{Pos}(\varphi)$.
Output: a pair $(\mathcal{C}, P')$, where $\mathcal{C}$ is a system of 0-1 ILP constraints, and $P' \subseteq P$ is the set of co-Gauss and atomic prime constituents occurring in $\pi(\varphi)$.

1. If $\pi$ is a position of a co-Gauss or atomic prime constituent in $P$, then
   1.1. Return $(\emptyset, \{\pi\})$.
2. If $\pi$ is a position of a Gauss prime constituent in $P$ or a leaf position, then
   2.1. Return $(\emptyset, \emptyset)$.
3. For each child position $\pi|1, \ldots, \pi|n$ of $\pi$ do
   3.1. $(\mathcal{C}_i, P_i') := \texttt{ILP-subroutine}(P, \varphi, x, \pi|i)$
4. $\mathcal{C} := \bigcup_{i=1}^n \mathcal{C}_i$
5. $P' := \bigcup_{i=1}^n P_i'$
6. If the top-level operator of $\pi(\varphi)$ is "$\vee$," then

6.1. Return $(\mathcal{C}, P')$.

7. If the top-level operator of $\pi(\varphi)$ is "$\wedge$," then

   7.1. For each pair of PCs $\mu_i \in P'_i$ and $\mu_j \in P'_j$ such that $i < j$, do

      7.1.1. Let $\pi_i$ and $\pi_j$ be the positions of $\mu_i$ and $\mu_j$, respectively.

      7.1.2. $\mathcal{C} := \mathcal{C} \cup \{x^l_{\pi_i} + x^u_{\pi_j} \geq 1,\, x^u_{\pi_i} + x^l_{\pi_j} \geq 1\}$

   7.2. Return $(\mathcal{C}, P')$.

It is easy to see that `ILP-construct` meets its specification and constructs system (3.1) for a given PC decomposition $P$ of $\varphi$. Its worst-case running time is quadratic in the size of the structural tree for $\varphi$ because of the loop in step 7 of `ILP-subroutine`. Consequently, algorithm`PC-bs-ILP` is correct as well.

### Reasonable Solutions of 0-1 ILP System $\mathcal{C}$

Algorithm `PC-bs-ILP` picks in step 2 *any* solution $s$ of the system $\mathcal{C}$. Therefore, it is correct to assign each variable the value 1 to obtain a solution of $\mathcal{C}$. However, it is of huge practical importance to obtain as small elimination set $E$ as possible. Here we propose a few strategies towards achieving this goal. These strategies are based on the observation that we have to be careful and minimize the number of variables that are assigned the value 1 in a solution $s$.

To do this one first constructs a linear objective function $\kappa$ in the variables $\bigcup_{\pi \in V} \{x^l_\pi, x^u_\pi\}$, where $V$ is the set of vertices of the graph $G(\varphi, P) = (V, E)$, i.e., the set of positions of all co-Gauss and atomic prime constituents in $P$. Minimizing $\kappa$ w.r.t. the system $\mathcal{C}$ then prevents unnecessary 1-assignments in $s$. A simple objective function $\kappa$ that ensures this is:

$$\kappa = \sum_{\pi \in V} x^l_\pi + x^u_\pi. \tag{3.2}$$

Here we point to the fact that it is straightforward to adjust `ILP-construct` to also return this objective function $\kappa$ for $P$ and to minimize $\kappa$ w.r.t. $\mathcal{C}$ in step 2 of `PC-bs-ILP` afterwards.

In practice one indeed wants to use an efficient 0-1 ILP solver [1] or a SAT solver with optimization functionality [40] for finding a solution of $\mathcal{C}$ with the smallest possible value of $\kappa$.

On the theoretical side observe that the facts that each constraint of (3.1) contains exactly two variables and that the objective function $\kappa$ from (3.2) is linear ensure that this optimization problem can be reduced to the minimum weighted vertex cover problem. This is a well-studied combinatorial optimization problem [4, 5, 57, 24]. However, it is possible that the special structure of the graph—that always comes to existence from $G(\varphi, P)$—allows for more efficient approximation and kernelization algorithms.

Now we return to our Example 65 where we showed three colorings $\chi_l$, $\chi_u$, and $\chi$. The values of the objective function $\kappa$ from (3.2) for these three colorings are 5, 5, and 6, respectively. Observe, however, that the cost 5 of $\chi_l$ does not really correspond with the produced structural test points: The prime constituent at position $\pi_4$ added one to the objective function value (because $s(x^l_{\pi_4}) = 1$ and $s(x^l_{\pi_4}) = 0$ in this case) even though it does not produce any test point.

Therefore, it can make sense to multiply the variables in (3.2) by weights $w_\pi^l$ and $w_\pi^u$, respectively. Intuitively, it should be then possible to estimate the costs of produced test points more precisely. Consequently, such an adjusted $\kappa$ should guide a solver towards such solutions of the system $\mathcal{C}$ that are expected to yield "nicer" elimination sets after `PC-to-TPs`.

As an example that it can make a significant difference for a prime constituent whether its lower bound candidate solutions or upper bound candidate solutions are used to obtain test points we consider the atomic formula $f \leq 0$, where $f = x^3 + ax^2 + bx + c$ and $a,\ b,\ c \in \mathbb{Z}[\mathbf{u}]$. For simplicity we do not use clustering. Since the leading coefficient of $f$ is positive, there are only four potential real 3-types of $f$. What these real types look like was discussed in Subsection 2.5.2. The following parametric root descriptions cover weak lower bounds of $\Phi(f \leq 0, \mathbf{a})$ for any parameter values $\mathbf{a} \in \mathbb{R}^m$:

$$\big(f, (3,2)\big), \big(f, (4,2)\big).$$

At the same time, parametric root descriptions covering weak upper bounds of $\Phi(f \leq 0, \mathbf{a})$ for any parameter values $\mathbf{a} \in \mathbb{R}^m$ are

$$\big(f, (1,1)\big), \big(f, (2,1)\big), \big(f, (3,1)\big), \big(f, (3,2)\big), \big(f, (4,1)\big), \big(f, (4,3)\big).$$

Recall that, for example, the root specification $(3,1)$ of $f$ represents the first (counting from the left to the right) real root of $f\langle\mathbf{a}\rangle$ whenever $f\langle\mathbf{a}\rangle$ is of real type $(-1,0,1,0,1)$. Assuming that the costs, i.e., the time and the length of the obtained formula, of substituting a parametric root description $\big(f, (t,r)\big)$ are the same regardless of the root specification $(t,r)$ one would set $w_\pi^l$ to 2 and $w_\pi^u$ to 6. Therefore, $x_\pi^l + x_\pi^u$ would be replaced with $2x_\pi^l + 6x_\pi^u$ in the objective function (3.2). Here $\pi$ is the position of the atomic prime constituent $f \leq 0$ in the formula $\varphi$.

For our Example 65 a weighted objective function $\kappa$ based on the approach we have just described would look as follows:

$$\kappa = x_{\pi_1}^l + x_{\pi_1}^u + x_{\pi_2}^l + 0 \cdot x_{\pi_2}^u + x_{\pi_3}^l + 0 \cdot x_{\pi_3}^u + 0 \cdot x_{\pi_4}^l + x_{\pi_4}^u + 0 \cdot x_{\pi_5}^l + x_{\pi_5}^u.$$

The costs of the three admissible colorings $\chi_l$, $\chi_u$, and $\chi$ with respect to $\kappa$ are then 3, 3, and 2, respectively.

One could go even further and analyze the degree of $f$, the degrees of the parameters occurring in $f$, the number of recursive lower-degree virtual substitutions and lengths of the virtual substitution formula schemes of Appendix A to adjust the weights $w_\pi^l$ and $w_\pi^u$ of prime constituents in $P$ even more precisely. However, in most cases counting the number of root specifications producing a test point, as we have shown above, should already yield a good approximation of the actual costs of a prime constituent when its lower/upper bound candidate solutions are taken.

### 3.4.2 A Structural Scheme with Bound Selection

Now we are ready to present and prove the correctness of a structural quantifier elimination scheme that employs the bound selection strategy described above. This boils down to integrating `PC-bs-ILP` into `svs-scheme`, which is straightforward. For completeness we present here the whole scheme obtained this way:

Algorithm `svs-scheme-bs`$(\varphi, x)$.
Input: a quantifier-free Tarski formula $\varphi(\mathbf{u}, x)$, which is an $\wedge$-$\vee$-combination of atomic formulas, a variable $x$.
Output: a quantifier-free Tarski formula $\psi(\mathbf{u})$ equivalent to $\exists x(\varphi)$ or FAILED.

1. $P :=$ `PC-decomposition`$(\varphi, x)$
2. If $P$ is FAILED, then return FAILED.
3. $P :=$ `PC-bs-ILP`$(P, \varphi, x)$
4. $E :=$ `PC-to-TPs`$(P, \varphi, x)$
5. $\psi :=$ false
6. For each structural test point $(e, \pi, \mathcal{F}) \in E$ do
   - 6.1. If $e$ is $(f, S)$, then
     - 6.1.1. $\gamma_e :=$ `guard`$\big((f, S), x\big)$
     - 6.1.2. Compute $\varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} (f, S)]$ by replacing each atom $g \varrho 0$ occurring in $\varphi_\pi^{\mathcal{F}}$ with quantifier-free formula `vs-at`$(g \varrho 0, (f, S), x)$.
     - 6.1.3. $\psi := \psi \vee \big(\gamma_e \wedge \varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} (f, S)]\big)$
   - 6.2. If $e$ is $(f, S) \pm \varepsilon$, then
     - 6.2.1. $\gamma_e :=$ `guard`$\big((f, S), x\big)$
     - 6.2.2. Compute $\varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} (f, S) \pm \varepsilon]$ by replacing each atom $g \varrho 0$ occurring in $\varphi_\pi^{\mathcal{F}}$ with quantifier-free formula `vs-at`$\big(g \varrho 0, (f, S) \pm \varepsilon, x\big)$.
     - 6.2.3. $\psi := \psi \vee \big(\gamma_e \wedge \varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} (f, S) \pm \varepsilon]\big)$
   - 6.3. If $e$ is $\pm\infty$, then
     - 6.3.1. Compute $\varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} \pm\infty]$ by replacing each atom $g \varrho 0$ occurring in $\varphi_\pi^{\mathcal{F}}$ with quantifier-free formula `vs-at`$(g \varrho 0, \pm\infty, x)$.
     - 6.3.2. $\psi := \psi \vee \varphi_\pi^{\mathcal{F}}[x \mathbin{/\!\!/} \pm\infty]$
7. Return $\psi$.

Similarly as `svs-scheme`, we first compute a PC decomposition of $\varphi$ in step 1. If this is successful, then we are able to continue and compute a quantifier-free equivalent of $\exists x \varphi$, otherwise we return FAILED. The loop in step 6 is almost identical with its counterpart in `svs-scheme`; here we only need to treat the case of both $\pm\varepsilon$, which was not necessary there because of the simple bound selection strategy of `svs-scheme`.

**Theorem 66** (Structural Quantifier Elimination Scheme with Bound Selection)**.** *Algorithm* `svs-scheme-bs` *meets its specification.*

*Proof.* Let $\mathbf{a} \in \mathbb{R}^m$ be arbitrary parameter values. We need to prove that $\mathbb{R} \models \psi(\mathbf{a})$ if and only if $\Phi(\varphi, \mathbf{a}) \neq \emptyset$.

If $\mathbf{a}$ satisfies the formula $\psi$ returned by `svs-scheme-bs`, then the proof of the fact that $\Phi(\varphi, \mathbf{a}) \neq \emptyset$ is essentially the same as in the proof of Theorem 60. Therefore, we do not repeat it here.

Now assume that the satisfying set $\Phi(\varphi, \mathbf{a})$ is nonempty. In the following we prove that $\mathbb{R} \models \psi(\mathbf{a})$, i.e., $\mathbf{a}$ satisfies at least one disjunct of $\psi$ returned by `svs-scheme-bs`. We proceed similarly as in the proof of Theorem 60.

If $E$ obtained in step 4 of `svs-scheme` is $\{(\pm\infty, (), \emptyset)\}$, then we deduce that $\mathbf{a}$ satisfies $\varphi[x /\!\!/ \pm\infty]$, which is the only disjunct of $\psi$ in this case. In the following we therefore assume that there exist PCs $\mu_1$ and $\mu_2 \in P$ with candidate solutions of types from $\{\mathrm{IP}, \mathrm{WUB}, \mathrm{SUB}\}$ and from $\{\mathrm{IP}, \mathrm{WLB}, \mathrm{SLB}\}$, respectively.

Let $\pi_1, \ldots, \pi_n$ be the positions of Gauss prime constituents in the order as found by `PC-decomposition` (cf. line 8 of that algorithm). Using Theorem 56 we obtain that $\varphi \longleftrightarrow \varphi_{\pi_1} \vee \varphi_{\pi_2}^{\{\pi_1\}} \vee \cdots \vee \varphi_{\pi_n}^{\{\pi_1,\ldots,\pi_{n-1}\}} \vee \varphi^{\{\pi_1,\ldots,\pi_n\}}$. We distinguish two cases.

In the first case we have $\Phi\big(\varphi_{\pi_i}^{\{\pi_1,\ldots,\pi_{i-1}\}}, \mathbf{a}\big) \neq \emptyset$ for some $i \in \{1, \ldots, n\}$. Similarly as in the proof of Theorem 60, it follows by Lemma 58 and Lemma 59 that $\mathbb{R} \models \gamma \wedge \big(\varphi_{\pi_i}^{\{\pi_1,\ldots,\pi_{i-1}\}}\big)[x /\!\!/ (f, S)](\mathbf{a})$ for some candidate solution $(f, S)$ originating from a candidate solution set for $\pi_i(\varphi)$. Since `PC-bs-ILP` assigns in steps 4–6 the bound choice $B$ of each Gauss prime constituent either to $\{l\}$ or to $\{u\}$, `PC-to-TPs` adds the structural test point $\big((f, S), \pi_i, \{\pi_1, \ldots, \pi_{i-1}\}\big)$ to $E$. Therefore, $\gamma \wedge \big(\varphi_{\pi_i}^{\{\pi_1,\ldots,\pi_{i-1}\}}\big)[x /\!\!/ (f, S)]$ is indeed a disjunct of $\psi$ constructed in step 6 of `svs-scheme-bs` satisfied by $\mathbf{a}$.

In the second case we have $\Phi(\varphi^{\{\pi_1,\ldots,\pi_n\}}, \mathbf{a}) \neq \emptyset$. For the sake of simplicity we denote $\varphi^{\{\pi_1,\ldots,\pi_n\}}$ by $\phi$ in the following. Let $\pi_{n+1}, \ldots, \pi_m$, $m \geq n$, be the positions of all non-Gauss prime constituents, i.e., prime constituents of type COGAUSS or AT, in $P$. Applying $(m - n)$ times Theorem 60 on $\phi$ and its positions $\pi_{n+1}, \ldots, \pi_m$ we obtain the following equivalence:

$$\phi \longleftrightarrow \phi_{\pi_{n+1}} \vee \phi_{\pi_{n+2}}^{\{\pi_{n+1}\}} \vee \cdots \vee \phi_{\pi_m}^{\{\pi_{n+1},\ldots,\pi_{m-1}\}} \vee \phi^{\{\pi_{n+1},\ldots,\pi_m\}}.$$

Since $\Phi(\varphi^{\{\pi_1,\ldots,\pi_n\}}, \mathbf{a}) \neq \emptyset$ is nonempty, there are two cases to consider:

1. In the first case we have $\Phi\big(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}, \mathbf{a}\big) \neq \emptyset$ for some index $j \in \{n+1, \ldots, m\}$. Since $\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}$ is an $\wedge$-$\vee$-combination of Tarski atoms obtained from $\varphi$, and all atoms at or under positions $\{\pi_1, \ldots, \pi_{j-1}\}$ in $\varphi$ were replaced with "false," the union of the candidate solution sets of PCs at positions $\{\pi_j, \ldots, \pi_m\}$ is—by Proposition 15 and induction—a set of candidate solutions for $\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}$.

   Let $s$ be the solution of system $\mathcal{C}$ computed in step 2 of `PC-bs-ILP`.

   Assume first that $s(x_{\pi_j}^l) = 1$, i.e., the PC at position $\pi_j$ was assigned in step 3.4 of `PC-bs-ILP` a bound choice $B$ such that $l \in B$. Observe that $\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}$ consists of a subset of prime constituents at positions $\{\pi_j, \ldots, \pi_m\}$ conjunctively associated with $\pi_j$—those co-Gauss and atomic prime constituents of $\varphi$ that were not replaced with "false." Therefore, the fact that $s$ is a solution of system $\mathcal{C}$—constructed as in (3.1)—ensures that $l \in B$ holds for each prime constituent occurring in $\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}$. Now there are three cases to consider:

   (a) The satisfying set $\Phi\big(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}, \mathbf{a}\big)$ is bounded from below, and contains its infimum $\beta$. Since $\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}$ is an $\wedge$-$\vee$-combination of atomic formulas, there exists a prime constituent $\mu$ at position $\pi$ such that $\beta$ is an isolated point or a weak lower bound of the satisfying set $\Phi\big(\pi(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}), \mathbf{a}\big)$. Therefore, there exists a candidate solution

$(f, S, \tau)$ of the formula $\pi(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}})$ such that $\beta = (f, S)\langle \mathbf{a} \rangle$ and $\tau$ is "IP" or "WLB." Recall that we have proven that $l \in B$ holds for the bound choice of $\mu$, `PC-to-TPs` added structural test point $\big((f, S), \pi, \{\pi_1, \ldots, \pi_n\}\big)$ to $E$. At the same time, $\beta = (f, S)\langle \mathbf{a} \rangle$ implies that $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$, and Theorem 25 ensures that $\mathbf{a}$ satisfies $\gamma \wedge \big(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}\big)[x \mathbin{/\!\!/} (f, S)]$.

Using the same structural arguments as in the proof of Theorem 60 one can show that $\mathbf{a}$ also satisfies the formula

$$\gamma \wedge \Delta\big(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}, \{\pi\}\big)[x \mathbin{/\!\!/} (f, S)].$$

Finally, using the fact that $\phi = \varphi^{\{\pi_1,\ldots,\pi_n\}}$ and that $\varphi$ is an $\wedge$-$\vee$-combination of atomic formulas we obtain that $\mathbf{a}$ satisfies the formula $\gamma \wedge \big(\varphi_{\pi}^{\{\pi_1,\ldots,\pi_n\}}\big)[x \mathbin{/\!\!/} (f, S)]$, which is a disjunct of the formula $\psi$ constructed during an iteration of the loop in step 6 of `svs-scheme-bs`.

(b) The satisfying set $\Phi\big(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}, \mathbf{a}\big)$ is bounded from below, but it does not contain its infimum $\beta$. Proceeding similarly as in the previous case, we obtain that there exists a prime constituent $\mu$ at position $\pi$ such that $\beta$ is an excluded point or a strict lower bound of the satisfying set $\Phi\big(\pi(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}), \mathbf{a}\big)$. Therefore, there exists a candidate solution $(f, S, \tau)$ of the formula $\pi(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}})$ such that $\beta = (f, S)\langle \mathbf{a} \rangle$ and $\tau$ is equal to "EP" or "SLB." Since we have shown above that $l \in B$ for the bound choice of $\mu$, `PC-to-TPs` ensures that $\big((f, S) + \varepsilon, \pi, \{\pi_1, \ldots, \pi_n\}\big) \in E$. At the same time, $\beta = (f, S)\langle \mathbf{a} \rangle$ implies that $\mathbf{a}$ satisfies a guard $\gamma$ of $(f, S)$, and Theorem 35 (i) ensures that $\mathbf{a}$ satisfies $\gamma \wedge \big(\phi_{\pi_j}^{\{\pi_1,\ldots,\pi_n\}}\big)[x \mathbin{/\!\!/} (f, S) + \varepsilon]$.

Again, using the same structural arguments as in the proof of Theorem 60 one can show that $\mathbf{a}$ also satisfies the formula

$$\gamma \wedge \Delta\big(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}, \{\pi\}\big)[x \mathbin{/\!\!/} (f, S) + \varepsilon].$$

In analogy with the previous case, we then finally obtain that $\mathbf{a}$ satisfies $\gamma \wedge \big(\varphi_{\pi}^{\{\pi_1,\ldots,\pi_n\}}\big)[x \mathbin{/\!\!/} (f, S) + \varepsilon]$, which is a disjunct of the formula $\psi$ returned by `svs-scheme-bs`.

(c) The satisfying set $\Phi\big(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}, \mathbf{a}\big)$ is unbounded from below. In this case we deduce that $\mathbb{R} \models \phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}[x \mathbin{/\!\!/} -\infty](\mathbf{a})$, which implies $\mathbb{R} \models \varphi[x \mathbin{/\!\!/} -\infty](\mathbf{a})$. Since we assume that there exists a prime constituent such that its bound choice $B$ contains $l$, `PC-to-TPs` adds $\big(-\infty, (), \emptyset\big)$ to $E$. Therefore, $\varphi[x \mathbin{/\!\!/} -\infty]$ is indeed a disjunct occurring in $\psi$ returned by `svs-scheme-bs`.

If the PC at position $\pi_j$ was assigned in `PC-bs-ILP` a bound choice $B$ such that $u \in B$, then the proof is done analogously using upper bounds, so we omit it. In either case one can show that $\mathbf{a}$ satisfies $\psi$ whenever $\Phi\big(\phi_{\pi_j}^{\{\pi_{n+1},\ldots,\pi_{j-1}\}}, \mathbf{a}\big) \neq \emptyset$.

2. In the second case we have $\Phi\big(\phi^{\{\pi_{n+1},\ldots,\pi_m\}}, \mathbf{a}\big) \neq \emptyset$. Observe that the formula $\phi^{\{\pi_{n+1},\ldots,\pi_m\}}$ does not contain the variable $x$, so the assumption

$\Phi\big(\phi^{\{\pi_{n+1},\ldots,\pi_m\}},\mathbf{a}\big) \neq \emptyset$ implies that $\Phi\big(\phi^{\{\pi_{n+1},\ldots,\pi_m\}},\mathbf{a}\big) = \mathbb{R}$. Thus, Theorem 35 (ii) ensures that $\mathbb{R} \models \phi^{\{\pi_{n+1},\ldots,\pi_m\}}[x /\!\!/ \pm\infty]$. Since $\phi^{\{\pi_{n+1},\ldots,\pi_m\}}$ is obtained from $\wedge$-$\vee$-combination $\varphi$ by replacing some of its atoms with "false" we also have $\mathbb{R} \models \varphi[x /\!\!/ \pm\infty]$. At the same time, we assume that there exists at least one prime constituent in $P$, and `PC-bs-ILP` guarantees by solving system $\mathcal{C}$ that for each prime constituent we have either $l \in B$ or $u \in B$ after step 3, so `PC-to-TPs` adds either $\big(-\infty,(),\emptyset\big)$ or $\big(\infty,(),\emptyset\big)$ to $E$. This finishes the proof of the fact that $\mathbf{a}$ satisfies $\psi$ for the case when $\Phi\big(\phi^{\{\pi_{n+1},\ldots,\pi_m\}},\mathbf{a}\big) \neq \emptyset$.

To sum up, we have just proven that $\mathbf{a}$ satisfies the formula $\psi$ returned by `svs-scheme-bs` whenever the satisfying set $\Phi(\varphi^{\{\pi_1,\ldots,\pi_n\}},\mathbf{a}) \neq \emptyset$ is nonempty. This finishes the proof of the theorem. □

Similarly as with `svs-scheme`, it can happen here that the same test point is generated more than once but with different positions and false replacement sets. To prevent repetitive substitutions of such a test point one can use conflation:

**Theorem 67** (Structural QE Scheme with Bound Selection and Conflation)**.** *Replace line 4 of* `svs-scheme-bs` *with*

$$E := \texttt{TPs-conflate}\big(\texttt{PC-to-TPs}(P,\varphi,x)\big).$$

*This modified structural quantifier elimination algorithm scheme with bound selection and conflation is correct.*

*Proof sketch.* The structure of the proof is the same as the structure of the proof of Theorem 66. The only difference is in the proof of the fact that $\mathbb{R} \models \exists x(\varphi)(\mathbf{a})$ implies $\mathbb{R} \models \psi(\mathbf{a})$ for $\psi$ returned by `svs-scheme-bs` with conflation.

When we argue that there exists a structural test point $(e,\pi,\mathcal{F})$ in the elimination set computed by `svs-scheme-bs` such that substituting it into some formula $\theta_\pi^{\mathcal{F}}$ yields a formula satisfied by $\mathbf{a}$ we use here the correctness of `TPs-conflate` to deduce that there exists a structural test point $(e,\pi',\mathcal{F}')$ in $E$ such that $\pi \sqsubseteq \pi'$ and $\mathcal{F}' \subseteq \mathcal{F}$. By $E$ we denote here the set returned by $\texttt{TPs-conflate}\big(\texttt{PC-to-TPs}(P,\varphi,x)\big)$. Proposition 53 then ensures that the substitution of $(e,\pi',\mathcal{F}')$ into $\theta_{\pi'}^{\mathcal{F}'}$ yields a formula satisfied by $\mathbf{a}$ as well. This observation suffices to adjust the proof of Theorem 66 to prove this theorem. □

Another similarity of `svs-scheme-bs` with `svs-scheme` is the possibility to use Theorem 19 (ii) to exclude all candidate solutions of type "EP" and include "WLB" and "WUB" candidate solutions instead in procedure `PC-to-TPs`. More specifically, replacing condition "$\tau \in \{\mathrm{EP},\mathrm{SLB}\}$" with condition "$\tau \in \{\mathrm{WLB},\mathrm{SLB}\}$" in step 4.1.2 and condition "$\tau \in \{\mathrm{EP},\mathrm{SUB}\}$" with condition "$\tau \in \{\mathrm{WUB},\mathrm{SUB}\}$" in step 4.1.4 of `PC-to-TPs` and using this modification in algorithm `svs-scheme-bs` (with conflation) yields a correct quantifier elimination algorithm scheme.

A difference between `svs-scheme-bs` and `svs-scheme`, in contrast, is that the scheme with bound selection is not a generalization of `vs-scheme` from Chapter 2 in the sense that a set of test points considered by `svs-scheme-bs` is a subset of a set of test points considered by `vs-scheme`; we simply cannot guarantee that when a structural test point $(e,\pi,\mathcal{F})$ is substituted by `svs-scheme-bs`

then $e$ is substituted by `vs-scheme` as well. The reason for this is that algorithm `svs-scheme-bs` possibly considers completely different elimination sets thank to solving a 0-1 ILP system of the form (3.1). However, for solutions of (3.1) that correspond to the global lower/upper bound selection this is indeed the case, because the set of test points considered by `svs-scheme-bs` is then identical with the set considered by `svs-scheme`.

Let us conclude the discussion of `svs-scheme-bs` with a few remarks on computation of the DNF, complexity, and the practical applicability of our bound selection algorithm.

Firstly, we would like to point to the following straightforward approach, which actually triggered our interest in bound selection strategies: After computing the DNF of $\varphi$ and pushing $\exists x$ inside the disjunction, one can make a separate "global" decision in each of the DNF members separately. Indeed, some parametric root descriptions will be considered more than once, but it is possible that in this way we obtain very simple and small elimination sets for the DNF members representing now completely independent real QE problems. The drawback of this straightforward approach is that computing the DNF before elimination of every single quantifier leads to an algorithm of non-elementary worst-case complexity in cases when the number of quantifiers is not constant in the length of the input formula.

Secondly, solving a 0-1 ILP instance despite its exponential worst-case complexity does not change the doubly exponential worst-case complexity of real quantifier elimination by virtual substitution. Therefore, our bound selection strategy is so to say "for free" when the worst-case complexity is considered.

Thirdly, we expect our bound selection strategy to have a reasonable practical potential: Since the global bound selection strategy, first presented in [85], was a giant step toward the practical applicability of real QE by virtual substitution, we strongly believe that our bound selection strategy based on 0-1 ILP should be a step forward from the practical point of view as well.

Finally, notice that our bound selection strategy is equivalent to the global bound selection strategy when applied to a pure conjunction of atomic formulas. At the same time, the elimination of a quantifier using virtual substitution introduces richer Boolean structure. This makes our bound selection strategy an appealing alternative to both the above-mentioned DNF computation as well as the global bound selection strategy; especially when more than one variable should be eliminated. The bound selection technique of this section can thus be intuitively described as a heuristic that: "does hopefully something useful even without computing the DNF," generalizes the global bound selection strategy, and does not increase the worst-case complexity of real QE by virtual substitution.

## 3.5 Conclusions

In this chapter we have studied the potential of exploiting the Boolean structure of an input formula $\varphi$ to boost our quantifier elimination scheme of Chapter 2. First we have introduced in Section 3.1 the notion of prime constituent decomposition of $\varphi$. We continued with the notions of conjunctive associativity, DNF, and the Marking technique in Section 3.2. The main result of this chapter is the structural quantifier elimination scheme presented and proven correct in Sec-

tion 3.3. Afterwards we discussed some extensions of the scheme like conflation and bound selection. The latter was the subject of Section 3.4.

The main lesson learned from our investigations in this chapter can be wrapped up as follows: It is helpful to analyze the Boolean structure during quantifier elimination. The overhead caused by prime constituent decomposition computation, conflation, and bound selection is worth the effort. Without bound selection we can even guarantee that we will never be worse as the approach of Chapter 2. With bound selection we possibly obtain simpler structural elimination sets but cannot guarantee this.

The notion of prime constituent—which is a generalization of the notion of atomic formula for the purposes of QE—allows us in some cases to reduce the size of an elimination set, prevent failing, and take advantage of equational constraints implied by subformulas of $\varphi$. The conjunctive associativity and condensing help us to obtain shorter resulting quantifier-free formulas. Moreover, our bound selection algorithm based on 0-1 ILP has the potential to yield even simpler structural elimination sets. Note also that all the techniques of this chapter have been integrated quite naturally into the theoretical framework of Chapter 2. On the practical side, this resulted in a robust framework that is extremely easy and straightforward to implement.

Finally, we point to the fact that the techniques and notions of PC decomposition, (co-)Gauss formulas, Marking, condensing, and bound selection are independent of the real closed field domain we work with in this thesis. It is an exciting future research direction to investigate their potential and possible generalizations in other domains that admit quantifier elimination.

# Chapter 4

# Degree Shift

In the previous chapters we have developed a framework for virtual substitution. We have seen that various enhancements like clustering, structural virtual substitution, and bound selection strategies nicely fit into the whole framework. In this chapter we proceed further in this direction: We study and integrate another technique called *degree shift* into our framework for virtual substitution.

Degree shift is a heuristic to reduce the degree of a quantified variable $x$. The technique was first mentioned in [84] and later implemented and successfully applied in the area of automated theorem proving in geometry [30]. The principal idea can be stated as follows: If a quantified variable $x$ occurs in an input formula $\varphi$ exclusively with powers divisible by $d > 1$, then it is possible to replace $x^d$ with $x$ in $\varphi$ to obtain an equivalent formula with strictly smaller maximum degrees of $x$. Despite its applicability only in special cases, it became clear that this technique is of great practical importance in the context of virtual substitution for the following reasons:

1. If applicable, a degree shift decreases the maximum degree of a quantified variable. Since virtual substitution comes with an upper degree bound on its applicability, a degree shift can possibly transform a formula for which virtual substitution fails (because of too high degrees of quantified variables) into an equivalent formula for which virtual substitution succeeds.

2. Even in cases when virtual substitution is directly applicable, it makes sense to investigate whether $\varphi$ allows for a degree shift beforehand. For example, it can happen that a quadratic or a cubic variable would be replaced by a linear one. This would be a twofold benefit: Firstly, a lower degree of the quantified variable yields a shorter quantifier-free equivalent. Secondly, the increase in the degrees of the other variables is in general smaller when the degree of the eliminated variable is smaller. Decreasing the degrees of the other variables in a quantifier-free equivalent can be of crucial importance for subsequent quantifier eliminations by virtual substitution.

3. Real-world quantifier elimination problems often exhibit structure that allows for numerous applications of degree shifts during the quantifier elimination process [30]. More importantly, without degree shifts some problems would not be solvable by virtual substitution at all.

To integrate degree shift into our framework we proceed in this chapter as follows: We first describe the original approach, which decreases the degree of $x$ when the GCD of all occurrences of $x$ in $\varphi$ is greater than one [30]. Here we refer to this approach as global degree shift. Building on our work [47], we then show how to perform a global degree shift by virtual substitution as we defined it in Chapter 2. We show that a global degree shift can be viewed as a virtual substitution of a parametric root description, and we provide concrete realizations of `guard` and `vs-prd-at` accordingly.

After this we study structural degree shift, which takes advantage of the Boolean structure of the input formula $\varphi$. Unfortunately, our construction will reveal that it is strongly related to the computation of the DNF (in the sense of Section 3.2) of $\varphi$ in the worst-case. Nevertheless, we discuss applicability and various extensions of structural degree shift in practice. Along the way we generalize the global degree shift and look closer at the connection between decreasing the maximum degree of a variable in $\varphi$ and the computation of the DNF of $\varphi$.

## 4.1 Global Degree Shift

The idea of *global degree shift* first appeared in [84, Section 3]. Later it was used as a heuristic to cope with higher than quadratic degrees in the context of quantifier elimination for formulas originating from the area of automated geometric theorem proving [30]. Since this technique is of general interest, and we build on it in the following, we restate the original idea here:

As usual let $\varphi$ be an $\wedge$-$\vee$-combination of Tarski atomic formulas. Let $d > 0$ be the GCD of all exponents of $x$ in $\varphi$. We divide all exponents of $x$ in $\varphi$ by $d$ obtaining $\varphi'$. If $d$ is odd, then we have $\exists x(\varphi) \longleftrightarrow \exists x(\varphi')$. If $d$ is even, then we have $\exists x(\varphi) \longleftrightarrow \exists x(x \geq 0 \wedge \varphi')$. For $d > 1$ this reduces the degree of $x$ in $\varphi$.

Notice that in order to obtain larger GCDs and hence a better degree reduction, we may in advance "adjust" the degree $k > 0$ of $x$ in an atomic formula of the form $cx^k \ \varrho \ 0$, where $x$ does not occur in $c$, using Proposition 22 as follows: In equations and negated equations, i.e., $\varrho \in \{=, \neq\}$, $k$ may be equivalently replaced with any $k' > 0$. In ordering inequalities, i.e., $\varrho \in \{<, \leq, \geq, >\}$, we may choose any $k' > 0$ of the same parity as $k$. During this adjustment one should in general use $k'$ that is as small as possible to keep the degree of $x$ in $\varphi'$ low.

In [47, Section 5] we have shown how to realize global degree shift by virtual substitution. This makes an integration of global degree shift into our framework of Chapter 2 even easier. Next we therefore explain and prove the correctness of the realization from [47], because we will build on it later.

Let $f \ \varrho \ 0$ be an atomic formula such that $f = c_k x^k + \cdots + c_1 x + c_0$, the coefficients $c_0, \ldots, c_k$ are elements of $\mathbb{Z}[\mathbf{u}]$, and $\varrho \in \{=, \neq, <, \leq, \geq, >\}$. As usual, $\mathbf{u} = u_0, \ldots, u_{m-1}$ are the parameters. Let $\hat{x}$ be a fresh variable, which does not occur in $f \ \varrho \ 0$. Let $d > 0$ be a natural number. We define the virtual substitution $[x /\!\!/ \sqrt[d]{\hat{x}}]$ of $\sqrt[d]{\hat{x}}$ for $x$ within atomic formula $f \ \varrho \ 0$ as follows:

$$\left(\sum_{j=0}^{k} c_j x^j \ \varrho \ 0\right)\left[x /\!\!/ \sqrt[d]{\hat{x}}\right] = \left(\sum_{j=0}^{k} c_j \hat{x}^{\lfloor \frac{j}{d} \rfloor} \ \varrho \ 0\right). \tag{4.1}$$

The floor function is applied to make the definition complete; we will *always* ensure that $d \mid j$ for every $j \in \{0, \dots, k\}$ when applying a degree shift by $d$ to atomic formula $\sum_{j=0}^{k} c_j x^j \ \varrho\ 0$. Note that the mapping $[x /\!\!/ \sqrt[d]{\hat{x}}]$ naturally generalizes from atomic to arbitrary quantifier-free formulas.

Now we are ready to prove the semantic correctness of global degree shift realized by virtual substitution as defined in (4.1):

**Proposition 68** (Correctness of Global Degree Shift)**.** *Let $\varphi(\mathbf{u}, x)$ be an $\wedge$-$\vee$-combination of Tarski atomic formulas. Let $d \geq 1$ be a divisor of all exponents of $x$ occurring in $\varphi$. Let $\hat{x}$ be a fresh variable that does not occur in $\varphi$. Then we have:*

*(i) If $d$ is even, then $\exists x(\varphi) \longleftrightarrow \exists \hat{x}\big(\hat{x} \geq 0 \wedge \varphi[x /\!\!/ \sqrt[d]{\hat{x}}]\big)$.*

*(ii) If $d$ is odd, then $\exists x(\varphi) \longleftrightarrow \exists \hat{x}\big(\varphi[x /\!\!/ \sqrt[d]{\hat{x}}]\big)$.*

*Proof.* (i) Assume that $d$ is even. Fix some real values $\mathbf{a} \in \mathbb{R}^m$ for the parameters $\mathbf{u}$. We show that we have

$$\mathbb{R} \models \Big(\exists x(\varphi) \longleftrightarrow \exists \hat{x}\big(\hat{x} \geq 0 \wedge \varphi[x /\!\!/ \sqrt[d]{\hat{x}}]\big)\Big)(\mathbf{a}).$$

First assume that there exists $b \in \mathbb{R}$ such that $(\mathbf{a}, b)$ satisfy $\varphi$. We show that $(\mathbf{a}, b^d)$ satisfy $\hat{x} \geq 0 \wedge \varphi[x /\!\!/ \sqrt[d]{\hat{x}}]$. Since $d$ is even, $b^d$ indeed satisfies $\hat{x} \geq 0$. Now it suffices to prove the following: The atomic formula $\hat{\alpha} = \sum_{j=0}^{k} c_j \hat{x}^{\lfloor \frac{j}{d} \rfloor} \ \varrho\ 0$ at position $\pi$ in $\varphi[x /\!\!/ \sqrt[d]{\hat{x}}]$—obtained from the atomic formula $\alpha = \sum_{j=0}^{k} c_j x^j \ \varrho\ 0$ at position $\pi$ in $\varphi$—is satisfied by $(\mathbf{a}, b^d)$ whenever $(\mathbf{a}, b)$ satisfy the atomic formula $\alpha$. The fact that $\varphi$ is an $\wedge$-$\vee$-combination of atoms will then imply that $(a, b^d)$ satisfy $\varphi[x /\!\!/ \sqrt[d]{\hat{x}}]$.

Since we assume that $d$ divides all exponents of $x$ occurring in $\varphi$, we obtain that $c_j = 0$ for every $j \in \{0, \dots, k\}$ such that $d \nmid j$. Therefore, we have

$$\sum_{j=0}^{k} c_j x^j = \sum_{\substack{0 \leq j \leq k \\ d \mid j}} c_j x^j \quad \text{and} \quad \sum_{j=0}^{k} c_j \hat{x}^{\lfloor \frac{j}{d} \rfloor} = \sum_{\substack{0 \leq j \leq k \\ d \mid j}} c_j \hat{x}^{\frac{j}{d}}. \tag{4.2}$$

This already ensures that whenever $(\mathbf{a}, b)$ satisfy the atomic formula $\alpha$, then $(\mathbf{a}, b^d)$ satisfy the atomic formula $\hat{\alpha}$.

Now assume that there exists $b \in \mathbb{R}$ such that $(\mathbf{a}, b)$ satisfy the formula $\hat{x} \geq 0 \wedge \varphi[x /\!\!/ \sqrt[d]{\hat{x}}]$. We show that $(\mathbf{a}, \sqrt[d]{b})$ satisfy $\varphi$. Since $b$ satisfies $\hat{x} \geq 0$, $\sqrt[d]{b} \in \mathbb{R}$. Similarly as above, it is sufficient to look at atomic formulas in $\varphi$ and their counterparts in $\varphi[x /\!\!/ \sqrt[d]{\hat{x}}]$. Using equations (4.2) again, we can show that whenever $(\mathbf{a}, b)$ satisfy $\sum_{j=0}^{k} c_j \hat{x}^{\lfloor \frac{j}{d} \rfloor} \ \varrho\ 0$—obtained from $\sum_{j=0}^{k} c_j x^j \ \varrho\ 0$—then $(\mathbf{a}, \sqrt[d]{b})$ satisfy $\sum_{j=0}^{k} c_j x^j \ \varrho\ 0$. Since we assume that $(\mathbf{a}, b)$ satisfy $\hat{x} \geq 0 \wedge \varphi[x /\!\!/ \sqrt[d]{\hat{x}}]$, this ensures that $(\mathbf{a}, \sqrt[d]{b})$ satisfy $\varphi$, so the proof of (i) is finished.

(ii) The proof is similar to the proof of (i), so we omit it. □

We illustrate the application of a global degree shift along with adjustments that make it possibly on a simple example.

**Example 69.** Consider the formula $\exists x(\varphi)$:

$$\exists x(u_0x^2 > 0 \wedge x^6 + u_1x^3 + u_0 < 0).$$

As such it does not allow for global degree shift, because the GCD of the degrees of $x$ in $\varphi$ is 1. Observe, however, that $\varphi$ is equivalent to

$$\exists x(u_0x^6 > 0 \wedge x^6 + u_1x^3 + u_0 < 0).$$

This formula allows a degree shift with $d = 3$. Proposition 68 then guarantees that this formula, and therefore also $\varphi$, is equivalent to

$$\exists \hat{x}(u_0\hat{x}^2 > 0 \wedge \hat{x}^2 + u_1\hat{x} + u_0 < 0).$$

Similarly, formula $\varphi = u_0x^{14} \geq 0 \wedge x^8 + u_1x^4 + u_0 \leq 0$ allows for global degree shift with $d = 2$. Equivalently adjusting the formula beforehand to $u_0x^{12} \geq 0 \wedge x^8 + u_1x^4 + u_0 \leq 0$ we see that a global degree shift with $d = 4$ is possible, so $\exists x(\varphi)$ is equivalent to $\exists \hat{x}(\hat{x} \geq 0 \wedge u_0\hat{x}^3 \geq 0 \wedge \hat{x}^2 + u_1\hat{x} + u_0 \leq 0)$. $\diamond$

To integrate global degree shift into our framework of Chapter 2 we show how to apply definitions and tools developed there to allow for objects like $\sqrt[q]{\hat{x}}$ to be substituted for $x$ into a quantifier-free formula by means of virtual substitution we have just defined and proven correct.

For this, we have to slightly generalize the framework by introducing *shadow quantifiers*. Recall that we are considering the elimination of $\exists x$ from the formula $\exists x\big(\varphi(\mathbf{u}, x)\big)$, where $\mathbf{u}$ are the parameters. As a first step we switch to the equivalent problem $\exists \hat{x} \exists x\big(\varphi(\mathbf{u}, x)\big)$, where the shadow variable $\hat{x}$ does not occur in $\{u_0, \ldots, u_{m-1}, x\}$, so $\hat{x}$ does not occur in $\varphi$ either. Proceeding in accordance with the framework of Chapter 2 to obtain a quantifier-free equivalent of $\exists \hat{x} \exists x(\varphi)$, the shadow quantifier $\exists \hat{x}$ imposes a trivial elimination problem that needs to be "solved" after eliminating $\exists x$ from $\exists x(\varphi)$.

Here notice that strictly following `vs-scheme` of Chapter 2 one would not simply drop the quantifier $\exists \hat{x}$ with the argument that the variable $\hat{x}$ does not occur in the quantifier-free equivalent of $\exists x(\varphi)$. The scheme would yield a trivial elimination set like $E = \{\pm\infty\}$. Moreover, notice that using $\emptyset$ as an elimination set $E$ in `vs-scheme` always yields "false," which is incorrect.

Observe that in contrast to all elimination sets studied so far we introduce here a variable $\hat{x}$ which was not present in $\varphi$ before. That variable is bound by shadow quantifier $\exists \hat{x}$. Intuitively, for the elimination of $\exists \hat{x} \exists x$ we switch from one hard plus one trivial elimination step to two nontrivial elimination steps. Semantically we view $\hat{x}$ during the elimination of $\exists x$ as a parameter.

The termination of quantifier elimination involving shadow quantifiers follows from the termination of the underlying quantifier elimination method plus the fact that we will always use only finitely many shadow quantifiers for each regular quantifier.

To keep the notation simple, we will in the sequel not formally introduce shadow quantifiers for all quantifiers. Instead, we will always silently assume their presence whenever we perform a degree shift.

Consider now the elimination of $\exists x$. Assume that we are in the situation of Proposition 68, and $d > 1$ is the GCD of the degrees of $x$ in $\varphi$. We use an elimination set that depends on the parity of $d$: $E = \big\{\sqrt[d]{\hat{x}}\big\}$, where $\sqrt[d]{\hat{x}}$ is a

*global degree shift test point* such that $\hat{x}$ is a shadow variable for $x$ and $d > 1$. The symbol "$\sqrt[d]{\hat{x}}$" is merely a shorthand notation for the respective parametric root description: If $d$ is odd, then it stands for parametric root description

$$(f, S) = \big(x^d - \hat{x}, ((-1, 0, 1), 1)\big) \quad \text{with a guard true.}$$

If $d$ is even, then it stands for parametric root description

$$(f, S) = \Big(x^d - \hat{x}, \big\{((1, 0, 1), 1), ((1, 0, -1, 0, 1), 2)\big\}\Big) \quad \text{with a guard } \hat{x} \geq 0.$$

Observe here that everything is semantically correct for $f = x^d - \hat{x}$. For any value $\mathbf{a} \in \mathbb{R}^{m+1}$ of the parameters $\mathbf{u}$ and $\hat{x}$ the following holds: $f\langle\mathbf{a}\rangle$ is of real type $(-1, 0, 1)$ when $d$ is odd, and $f\langle\mathbf{a}\rangle$ is of real type $(1, 0, 1)$ or $(1, 0, -1, 0, 1)$ if and only if $\mathbb{R} \models (\hat{x} \geq 0)(\mathbf{a})$. Furthermore, Proposition 68 ensures the semantic correctness of the virtual substitution of $\sqrt[d]{\hat{x}}$ into an atomic formula $g \ \varrho \ 0$ whenever $d$ divides all the degrees of $x$ in $g$. In such a case it is obvious that (4.1) yields a formula with the following property: Whenever $\mathbf{a}$ satisfies a guard of $\sqrt[d]{\hat{x}}$, then $\mathbf{a}$ satisfies $(g \ \varrho \ 0)[x \mathbin{/\!\!/} \sqrt[d]{\hat{x}}]$ if and only if the formula $g \ \varrho \ 0$ holds at $(f, S)\langle\mathbf{a}\rangle$, which is indeed a well-defined real number. Therefore, using the above-mentioned guards along with (4.1) give us correct realizations of `guard` and `vs-prd-at`, respectively, for $\sqrt[d]{\hat{x}}$.

## 4.2 Structural Degree Shift

In this section we present an approach that performs a kind of structural degree shift. Again, we aim at exploiting the Boolean structure of the input formula. The approach presented here makes use of condensing and the Marking technique that we already applied in the context of structural virtual substitution in Chapter 3. We generalize global degree shift by looking only at conjunctively associated formulas during the GCD computation. The main idea of our approach is best illustrated with an example.

**Example 70.** Consider a quantifier-free Tarski formula $\varphi(a, b, x)$ in Figure 4.1. Since the GCD of all exponents of $x$ in $\varphi$ is 1, no global degree shift for $\varphi$ and $x$ is applicable. At the same time, observe that condensing $\varphi$ w.r.t. positions $\pi_3$, $\pi_4$, and $\pi_5$, respectively, and adjusting the degrees of $x$ in atomic formulas $bx \neq 0$ and $ax \geq 0$ when necessary we obtain:

$$\begin{aligned}
\varphi_{\pi_3} &\longleftrightarrow a(b-1)x^6 - b \geq 0 \wedge bx^3 \neq 0 \wedge x^3 + b \geq 0,\\
\varphi_{\pi_4} &\longleftrightarrow a(b-1)x^6 - b \geq 0 \wedge bx^2 \neq 0 \wedge x^2 - a \leq 0,\\
\varphi_{\pi_5} &\longleftrightarrow a(b-1)x^6 - b \geq 0 \wedge bx^3 \neq 0 \wedge ax^3 \geq 0.
\end{aligned}$$

Realizing a global degree shift in these equivalents of $\varphi_{\pi_3}$, $\varphi_{\pi_4}$, and $\varphi_{\pi_5}$ by $d = 3$, 2, and 3, respectively, we obtain:

$$\begin{aligned}
\exists x(\varphi_{\pi_3}) &\longleftrightarrow \exists\hat{x}\big(a(b-1)\hat{x}^2 - b \geq 0 \wedge b\hat{x} \neq 0 \wedge \hat{x} + b \geq 0\big),\\
\exists x(\varphi_{\pi_4}) &\longleftrightarrow \exists\hat{x}\big(\hat{x} \geq 0 \wedge a(b-1)\hat{x}^3 - b \geq 0 \wedge b\hat{x} \neq 0 \wedge \hat{x} - a \leq 0\big),\\
\exists x(\varphi_{\pi_5}) &\longleftrightarrow \exists\hat{x}\big(a(b-1)\hat{x}^2 - b \geq 0 \wedge b\hat{x} \neq 0 \wedge a\hat{x} \geq 0\big).
\end{aligned}$$

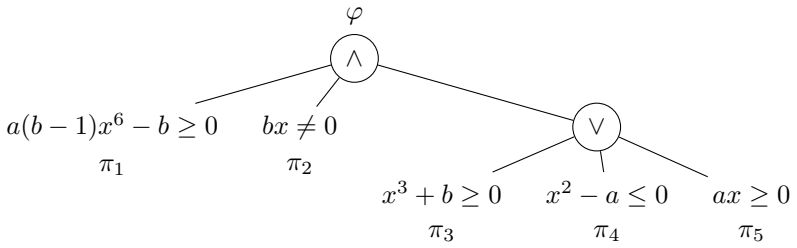



Figure 4.1: The structural tree for $\varphi(a, b, x)$ from Example 70. Each node is labeled with its position in the tree.

Applying the Marking technique (Theorem 56) on positions $\{\pi_3, \pi_4, \pi_5\}$ we obtain that $\varphi$ is equivalent to $\varphi_{\pi_3} \vee \varphi_{\pi_4} \vee \varphi_{\pi_5} \vee \varphi^{\{\pi_3,\pi_4,\pi_5\}}$. Since the degree of $\hat{x}$ is at most three in the respective equivalents of $\exists x(\varphi_{\pi_3})$, $\exists x(\varphi_{\pi_4})$, and $\exists x(\varphi_{\pi_5})$, we can use `vs-scheme` of Chapter 2 to successfully eliminate the quantifier $\exists \hat{x}$ from these equivalents. Finally, $\varphi^{\{\pi_3,\pi_4,\pi_5\}}$ is equivalent to "false," so the disjunction of these equivalents yields a quantifier-free equivalent of $\exists x(\varphi)$. $\diamond$

To take advantage of situations as the one shown in Example 70 we develop in this section a technique called structural degree shift.

Similarly as with other structural test points in Chapter 3, we define here a *structural degree shift test point* as a triple $\left(\sqrt[d]{\hat{x}}, \pi, \mathcal{F}\right)$, where $d \geq 1$, $\pi$ is a position in $\varphi$, and $\mathcal{F}$ is a false replacement set. Consequently, $\left(\sqrt[d]{\hat{x}}, (), \emptyset\right)$ is a structural degree shift test point representing the global degree shift by $d$. Notice that when $d = 1$ the substitution of a test point $\left(\sqrt[d]{\hat{x}}, \pi, \mathcal{F}\right)$ performs only the renaming of $x$ to $\hat{x}$ in the formula $\varphi_\pi^{\mathcal{F}}$. Therefore, we will *always* ensure that $d > 1$ for structural degree shift test points we produce.

The algorithm `s-shift-at` below computes a set of structural degree shift test points. Similarly as in Example 70, the algorithm condenses $\varphi$ w.r.t. each atomic position $\pi$ in $\varphi$ to determine where we could take advantage of a structural degree shift.

Observe that $\emptyset$ is returned when no structural degree shift w.r.t. an atomic position is possible for $\varphi$. For formula $\varphi$ from Example 70 `s-shift-at`$(\varphi, x)$ returns the set $S = \left\{\left(\sqrt[3]{x}, \pi_3, \emptyset\right), \left(\sqrt[2]{x}, \pi_4, \emptyset\right), \left(\sqrt[3]{x}, \pi_5, \emptyset\right)\right\}$.

Algorithm `s-shift-at`$(\varphi, x)$.
Input: an $\wedge$-$\vee$-combination of Tarski atomic formulas $\varphi$, a variable $x$.
Output: a set $S$ of structural degree shift test points.

1. $S := \emptyset$
2. For each atomic position $\pi \in$ `Pos`$(\varphi)$ do
   - 2.1. Compute the GCD $d$ of all occurrences of $x$ in the formula $\varphi_\pi$, adjusting the degree of $x$ in atoms of the form $cx^k \varrho\, 0$ where necessary.
   - 2.2. If $d > 1$, then
     - 2.2.1. Add $\left(\sqrt[d]{\hat{x}}, \pi, \emptyset\right)$ to $S$.
3. Return $S$.

Since `s-shift-at` computes the GCD of $\varphi_\pi$ for every atomic position $\pi \in \mathtt{Pos}(\varphi)$, the algorithm needs to call a routine computing the GCD of a pair of integers at most $O\big(|\varphi| \log(|\varphi|)\big)$ times. Algorithm `s-shift-at` obviously meets its specification. Next we prove that it is correct to use the algorithm as a preprocessing step in the context of quantifier elimination:

**Theorem 71.** *Let* $S = \Big\{ \big( \sqrt[d_1]{\hat{x}}, \pi_1, \emptyset \big), \ldots, \big( \sqrt[d_n]{\hat{x}}, \pi_n, \emptyset \big) \Big\}$ *be the set of structural degree shift test points returned by* `s-shift-at`$(\varphi, x)$. *Then* $\exists x(\varphi)$ *is equivalent to*

$$\exists \hat{x} \big( \gamma_1 \wedge \varphi_{\pi_1}[x \mathbin{/\!\!/} \sqrt[d_1]{\hat{x}}] \big) \vee \cdots \vee \exists \hat{x} \big( \gamma_n \wedge \varphi_{\pi_n}[x \mathbin{/\!\!/} \sqrt[d_n]{\hat{x}}] \big) \vee \exists x \big( \varphi^{\{\pi_1, \ldots, \pi_n\}} \big), \quad (4.3)$$

*where* $\gamma_i$ *is a guard of* $\big( \sqrt[d_i]{\hat{x}}, \pi_i, \emptyset \big)$. *We assume here that in* $\varphi_{\pi_i}$ *the same adjustments as in step 2.1 of* `s-shift-at` *took place before substituting* $\big( \sqrt[d_i]{\hat{x}}, \pi_i, \emptyset \big)$ *for every* $i \in \{1, \ldots, n\}$.

*Proof.* W.l.o.g. we assume that the atomic positions $\pi_1, \ldots, \pi_n$ are in the same order as they were added to $S$ in step 2.2.1 of `s-shift-at`. Since these positions are pairwise independent, applying Marking (Theorem 56) to $\{\pi_1, \ldots, \pi_n\}$ guarantees that $\varphi$ is equivalent to $\varphi_{\pi_1} \vee \varphi_{\pi_2} \vee \cdots \vee \varphi_{\pi_n} \vee \varphi^{\{\pi_1, \ldots, \pi_n\}}$. Observe that $d_i > 1$ is—eventually after some degree adjustments of $\varphi_{\pi_i}$'s atomic subformulas of the form $cx^k \mathrel{\varrho} 0$—the GCD of all $x$-exponents in $\varphi_{\pi_i}$, so $d_i$ divides the degree of each $x$-occurrence in $\varphi_{\pi_i}$ as well. Therefore, Proposition 68 assures us that $\exists x(\varphi_{\pi_i})$ is equivalent to $\exists \hat{x} \big( \gamma_i \wedge \varphi_{\pi_i}[x \mathbin{/\!\!/} \sqrt[d_i]{\hat{x}}] \big)$, because $\gamma_i$ is the formula "true" when $d_i$ is odd and the formula "$\hat{x} \geq 0$" when $d_i$ is even. This already shows that $\exists x(\varphi)$ is equivalent to the formula (4.3). □

Observe that in practice one needs to store with a test point $\big( \sqrt[d_i]{\hat{x}}, \pi_i, \emptyset \big)$ also the adjustments done in step 2.1 of `s-shift-at`, i.e., for each $\varphi_{\pi_i}$'s atomic subformula of the form $cx^k \mathrel{\varrho} 0$ one needs to remember $k'$ that equivalently replaced $k$ to ensure that $d_i \mid k'$. Only after this adjustment—which is done tacitly in (4.3)—we can apply Proposition 68.

The maximum degree of $x$ and $\hat{x}$ in the formula (4.3) is guaranteed to be strictly smaller than the maximum degree of $x$ in $\varphi$ whenever $\varphi^{\{\pi_1, \ldots, \pi_n\}}$ does not contain $x$. This is for example the case when $\varphi^{\{\pi_1, \ldots, \pi_n\}}$ simplifies to "false."

A natural possibility to integrate `s-shift-at` into `svs-scheme` is to use it as a preprocessing before calling `svs-scheme`. This is done by the recursive algorithm `s-preproc-at` below. The algorithm uses Marking and tries until no degree shift is applicable anymore. In this way we try to take advantage of structural shifts as much as possible. The correctness of `s-preproc-at` follows directly from Theorem 71. Consequently, to obtain a quantifier-free equivalent of $\exists x(\varphi)$, one applies `svs-scheme` to each element of the set returned by `s-preproc-at`$(\varphi, x)$.

Algorithm `s-preproc-at`$(\varphi, x)$.
Input: an $\wedge$-$\vee$-combination of atomic formulas $\varphi$, a variable $x$.
Output: a finite set of pairs $\big\{(\varphi_i, x_i)\big\}_i$ such that $\varphi_i$ is an $\wedge$-$\vee$-combination of Tarski atoms, $x_i$ is a variable, and $\exists x(\varphi)$ is equivalent to $\bigvee_i \exists x_i(\varphi_i)$.

1. $S :=$ `s-shift-at`$(\varphi, x)$
2. If $S = \emptyset$, then

2.1. Return $\{(\varphi, x)\}$.

3. $T := \emptyset$
4. Let $S$ be $\Big\{\big(\sqrt[d_1]{\hat{x}}, \pi_1, \emptyset\big), \ldots, \big(\sqrt[d_n]{\hat{x}}, \pi_n, \emptyset\big)\Big\}$. For $i := 1$ to $n$ do

   4.1. $T := T \cup \texttt{s-preproc-at}\big(\gamma_i \wedge \varphi_{\pi_i}[x \mathbin{/\!\!/} \sqrt[d_i]{\hat{x}}], \hat{x}\big)$
5. $T := T \cup \texttt{s-preproc-at}(\varphi^{\{\pi_1,\ldots,\pi_n\}}, x)$
6. Return $T$.

Observe that repeated applications of subroutine `s-shift-at` within recursive calls to `s-preproc-at` implicitly use more than one shadow variable for $x$. It is correct to use one or even no shadow quantifiers at all. In this way we would be carrying out only equivalent transformations on $\exists x(\varphi)$. If one is interested only in obtaining a quantifier-free equivalent of $\exists x(\varphi)$, this is indeed a correct approach. However, tracking the history of used shadow quantifiers and the relation of shadow variables to the original "non-shadow" variable $x$ will be of crucial importance in Chapter 5. There we will consider existential sentences and show how to compute concrete real values for each quantified variable whenever an input sentence is satisfiable.

### 4.2.1 A Lower Bound for `s-preproc-at`

In the following we show how to construct for any $h \in \mathbb{N} \setminus \{0\}$ a Tarski formula $\phi$ such that $\texttt{s-preproc-at}(\phi, x)$ eventually computes each disjunct of the DNF of $\phi$. In our construction we focus mainly on the degrees of $x$ in the atomic formulas of $\phi$. We begin by constructing a "bad" *degree formula* $\delta$. From the bad degree formula we then construct a Tarski formula $\phi(a, b, x)$ such that $\texttt{s-preproc-at}(\phi, x)$ eventually computes the DNF of $\phi$.

For a quantifier-free Tarski formula $\varphi$ we obtain its degree formula $\delta$ by simply replacing each atomic formula in $\varphi$ with the GCD of the degrees of $x$ occurring in it. For a degree formula we will use the terms "atomic subformula" and "integer" interchangeably, because the atoms of degree formulas are just non-negative integers. Therefore, $\texttt{At}(\delta)$ is a set of integers occurring in $\delta$. For a formula $\varphi$ and its degree-formula $\delta$ we define $\gcd(\varphi) = \gcd(\delta)$ to be the GCD of all degrees of $x$-occurrences in $\texttt{At}(\varphi)$. These definitions are illustrated in Figure 4.2. Notice that concepts like condensing, deletion, and Marking carry over naturally to degree formulas, so we will directly use these along with their established notations without explicitly defining them for degree formulas.

To begin our construction, we fix $h \in \mathbb{N} \setminus \{0\}$. Before constructing $\delta$ we first construct a degree formula $\overline{\delta}$ of height $2h$ meeting the following specification:

1. Every inner node of $\overline{\delta}$ has exactly two children.
2. An inner node of $\overline{\delta}$ with an odd distance from the root is an $\vee$-node.
3. An inner node of $\overline{\delta}$ with an even distance from the root is an $\wedge$-node.
4. The leaf positions of $\overline{\delta}$ are pairwise distinct odd primes. We denote the prime at an atomic position $\alpha$ by $p_\alpha$.

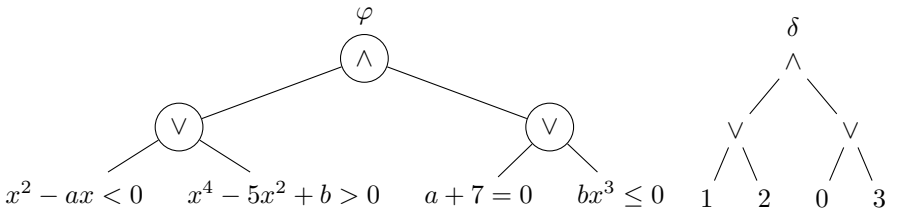



Figure 4.2: A Tarski formula $\varphi$ along with its degree formula $\delta$. Observe that we have $\gcd(\varphi) = \gcd(\delta) = 1$ and $\gcd\big((2)(\varphi)\big) = \gcd\big((2)(\delta)\big) = 3$.

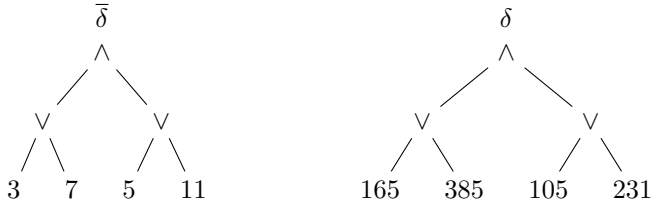



Figure 4.3: An example degree formula $\overline{\delta}$ meeting specification 1–4 for $h = 1$ and the degree formula $\delta$ constructed from $\overline{\delta}$ using our construction. For $\delta$ we have: $\gcd(\delta_{(1,1)}) = 3$, $\gcd(\delta_{(1,2)}) = 7$, $\gcd(\delta_{(2,1)}) = 5$, and $\gcd(\delta_{(2,2)}) = 11$.

Using $\overline{\delta}$ we then construct the degree formula $\delta$ of height $2h$ with the same Boolean structure as $\overline{\delta}$ as follows: Each atomic position $\alpha$ of $\delta$ is the product of the prime at position $\alpha$ and primes at atomic positions of $\overline{\delta}$ that are conjunctively associated with $\alpha$. An example of this construction for $h = 1$ is shown in Figure 4.3. Next we prove a few properties of $\delta$ that will be used for proving our lower bound later:

**Lemma 72.** *Let $h \in \mathbb{N} \setminus \{0\}$. Let $\overline{\delta}$ and $\delta$ be degree formulas of height $2h$ whose construction has been described above. Let $\alpha$ be an atomic position of $\delta$. Let $\pi$ be a position of $\delta$ such that $\pi \neq \alpha$. Then $\pi \curlywedge \alpha$ if and only if $p_\alpha$ divides each integer in $\mathtt{At}(\pi(\delta))$.*

*Proof.* If $\pi$ is an atomic position, then (i) is directly implied by the construction of $\delta$. Next we assume that $\pi$ is an inner position.

First we assume that $\pi \curlywedge \alpha$. This means that every atomic position under $\pi$ is conjunctively associated with $\alpha$ as well. The construction of $\delta$ now ensures that $p_\alpha$ divides each integer in $\mathtt{At}(\pi(\delta))$.

To prove the converse assume that $p_\alpha$ divides each integer in $\mathtt{At}(\pi(\delta))$. The construction of $\delta$ ensures that each atomic position under $\pi$ is conjunctively associated with $\alpha$, i.e., the lowest common ancestor of each atomic position under $\pi$ and $\alpha$ is an $\wedge$-node. This ensures that $\pi$ and $\alpha$ are independent and that $\mathrm{lca}(\pi, \alpha)$ is an $\wedge$-node, i.e., $\pi \curlywedge \alpha$. □

**Lemma 73.** *Let $h \in \mathbb{N} \setminus \{0\}$. Let $\overline{\delta}$ and $\delta$ be degree formulas of height $2h$ whose construction has been described above. Then the following hold:*

*(i) If $\alpha$ is an atomic position of $\delta$, then $\gcd(\delta_\alpha) = p_\alpha$.*

*(ii) If $\pi$ is an inner position of $\delta$, then $\gcd(\delta_\pi) = 1$.*

*Proof.* (i) Let $\alpha$ be an atomic position of $\delta$. Observe that the formula $\delta_\alpha = \Gamma(\delta, \alpha)$ consists of the atomic formula at position $\alpha$ and other atomic formulas, all of which are either conjunctively associated with $\alpha$ or were replaced with 0 to obtain $\delta_\alpha$ from $\delta$. This ensures that $p_\alpha$ divides each integer in $\mathtt{At}(\delta_\alpha)$, i.e., $\gcd(\delta_\alpha) \geq p_\alpha$. Next we show that $\gcd(\delta_\alpha) \leq p_\alpha$. We assume for a contradiction that $\gcd(\delta_\alpha) > p_\alpha$.

Notice that each integer in $\mathtt{At}(\delta)$ is square-free, and there exists a one-to-one correspondence between prime factors of $\mathtt{At}(\delta)$ and atomic positions of $\delta$. Therefore, the assumption $\gcd(\delta_\alpha) > p_\alpha$ together with the construction of $\delta$ ensure that there exists an atomic position $\alpha'$ such that $\alpha' \neq \alpha$, and $p_{\alpha'}$ divides each integer in $\mathtt{At}(\delta_\alpha)$.

Using Lemma 72 we obtain that $\alpha'$ is conjunctively associated with or equal to every atomic position—that was not replaced with 0—in $\delta_\alpha$. In particular, $\alpha' \curlywedge \alpha$, i.e., $\mathrm{lca}(\alpha, \alpha')$ is an $\wedge$-node. Recall that $\alpha'$ is an atomic position, the height of $\delta$ is even, and all $\wedge$-nodes have an even distance from the root of $\delta$. These facts ensure that there exists at least one $\vee$-node lying on the path from $\mathrm{lca}(\alpha, \alpha')$ to $\alpha'$. Now $\alpha'$ is conjunctively associated with no atomic position lying under this $\vee$-node in the branch not containing $\alpha'$. This contradicts the fact that $\alpha'$ is conjunctively associated with every atom of $\delta_\alpha$, because each atomic position in the branch not containing $\alpha'$ is conjunctively associated with $\alpha$—the lowest common ancestor of such a position and $\alpha$ is namely $\mathrm{lca}(\alpha, \alpha')$, which is an $\wedge$-node. This finishes the proof of (i).

(ii) Let $\pi$ be an inner position of $\delta$, and assume for a contradiction that $\gcd(\delta_\pi) > 1$. Similarly as in the proof of (i), it follows that there exists an atomic position $\alpha$ such that $p_\alpha$ divides each integer in $\mathtt{At}(\delta_\pi)$. Again, Lemma 72 assures us that $\alpha$ is conjunctively associated with every atomic position in $\delta_\pi$ that was not replaced with 0. Using the same argument as in the proof of (i) we obtain that there exists an atomic position $\alpha'$ that was not replaced with 0 in $\delta_\pi$ such that $\alpha' \neq \alpha$, and $\mathrm{lca}(\alpha', \pi)$ is an $\wedge$-node. Consequently, we deduce that $\alpha$ is not conjunctively associated with any atom lying under an $\vee$-node in the branch not containing $\alpha'$. This contradicts the fact that that $\alpha$ is conjunctively associated with every nonzero atom in $\delta(\pi)$. This shows that (ii) holds. □

**Lemma 74.** *Let $h \in \mathbb{N} \setminus \{0\}$. Let $\overline{\delta}$ and $\delta$ be degree formulas of height $2h$ whose construction has been described above. Let $\alpha_1, \ldots, \alpha_n$ be $n$ pairwise conjunctively associated atomic positions in $\delta$. Put $\delta_0 = \delta$ and for $i \in \{1, \ldots, n\}$ construct $\delta_i$ from $\delta_{i-1}$ as follows:*

1. *Replace in $\delta_{i-1}$ each integer at all atomic positions independent from $\alpha_i$ and not conjunctively associated with $\alpha_i$ with zero.*

2. *Divide each atomic position in the obtained formula by $p_{\alpha_i}$.*

*Let $\alpha$ be an atomic position of a nonzero integer. If $\alpha \in \{\alpha_1, \ldots, \alpha_n\}$, then $\gcd(\Gamma(\delta_n, \alpha)) = 1$. If $\alpha \notin \{\alpha_1, \ldots, \alpha_n\}$, then $\gcd(\Gamma(\delta_n, \alpha)) = p_\alpha$.*

*Proof.* To begin with, observe that when $n = 0$ the lemma follows directly from Lemma 73; for every atomic position $\alpha$ in $\delta_0$ we have $\gcd(\Gamma(\delta_0, \alpha)) = p_\alpha$.

Let $n > 0$ and assume that the lemma holds for $n-1$. Notice that the equivalency and GCD preserving transformation in Figure 4.4 transforms both Tarski and degree formulas to semantically equivalent formulas with the same GCD of degrees of all occurrences of a variable. Observe that the output DNF computed by $\texttt{the-dnf}\big(\delta_{n-1},()\big)$ can be obtained from $\delta_{n-1}$ by finitely many applications of the transformation.

Using Theorem 55 we therefore obtain that for any atomic position $\beta$ we have: The formula $\Gamma(\delta_{n-1},\beta)$ is equivalent to the disjunction of those disjuncts returned by $\texttt{the-dnf}\big(\delta_{n-1},()\big)$ that contain the integer originating from position $\beta$. Consequently, the GCD of $\Gamma(\delta_{n-1},\beta)$ and the GCD of these "$\beta$-disjuncts" are equal. The latter GCD can be computed by computing the GCD of all disjunction members after replacing each atomic formula independent from and not conjunctively associated with $\beta$ with zero, because then $p_\beta$ divides each and every atomic formula in the DNF of $\delta_{n-1}$.

Let now $\beta_1$, $\beta_2$ be two distinct nonzero atomic positions in $\delta_{n-1}$. We define

$$\begin{aligned} g_1 &= \gcd\big(\Gamma(\delta_{n-1},\beta_1)\big),\\ g_2 &= \gcd\big(\Gamma(\delta_{n-1},\beta_2)\big),\\ g_{1,2} &= \gcd\big(\Gamma(\Gamma(\delta_{n-1},\beta_1),\beta_2)\big). \end{aligned}$$

Using Theorem 55 again we obtain that $g_{1,2}$ is equal to the GCD of those disjuncts of $\texttt{the-dnf}\big(\delta_{n-1},()\big)$ that contain *both* $\beta_1$ and $\beta_2$.

Since the sets of DNF members containing $\beta_1$ and $\beta_2$, respectively, are obviously supersets of the set of DNF members containing both $\beta_1$ and $\beta_2$ we have $g_1 \mid g_{1,2}$ and $g_2 \mid g_{1,2}$. To prove that $g_{1,2} = g_1 g_2$ we next show that $g_{1,2} \mid g_1 g_2$.

Assume for a contradiction that $g_{1,2} \nmid g_1 g_2$. The induction hypothesis along with the construction of $\delta$ then imply that there exists $\beta \notin \{\beta_1,\beta_2\}$ such that $p_\beta$ divides $g_{1,2}$. Recall that $g_{1,2}$ is the GCD of those disjuncts of $\texttt{the-dnf}\big(\delta_{n-1},()\big)$ that contain both $\beta_1$ and $\beta_2$. The assumption $g_{1,2} \nmid g_1 g_2$ implies means that each DNF member in the set defined as the set of DNF members containing both $\beta_1$ and $\beta_2$ necessarily contains $\beta \notin \{\beta_1,\beta_2\}$, which is obviously not the case for our construction of $\delta$; a contradiction.

The process of obtaining $\delta_n$ from $\delta_{n-1}$ can be simulated on the DNF of $\delta_{n-1}$ by deleting the DNF members that do not contain $\alpha_n$, and dividing each atomic formula in the remaining members by $p_{\alpha_n}$. Observe that no $\alpha_i$ was replaced with zero during this process, because $\{\alpha_1,\ldots,\alpha_n\}$ are pairwise conjunctively associated. These facts ensure that $\gcd\big(\Gamma(\delta_{n-1},\alpha_n)\big)/p_{\alpha_n} = \gcd(\Gamma(\delta_n,\alpha_n))$. The induction hypothesis guarantees that $\gcd\big(\Gamma(\delta_{n-1},\alpha_n)\big) = p_{\alpha_n}$, so $\gcd\big(\Gamma(\delta_n,\alpha_n)\big) = 1$. This shows that $\gcd(\Gamma(\delta_n,\alpha)) = 1$ for any nonzero $\alpha \in \{\alpha_1,\ldots,\alpha_n\}$.

Finally, since we constructed $\overline{\delta}$ using pairwise distinct primes, the induction hypothesis together with the fact that we carried out a division by $p_{\alpha_n}$ ensures that $\gcd(\Gamma(\delta_n,\alpha)) = p_\alpha$ for any nonzero $\alpha \notin \{\alpha_1,\ldots,\alpha_n\}$. This finishes the induction step and also the proof of the lemma. □

With these properties of the degree formula $\delta$ at hand, we are now ready to construct the Tarski formula $\phi(b,c,x)$ such that `s-preproc-at` eventually constructs the DNF of $\phi$. Given a positive integer $h$ and the degree formula $\delta$ constructed for $h$, we construct $\phi(b,c,x)$ as follows: Replace each atomic position in $\delta$ containing integer $d$ with Tarski atomic formula $bx^d + c \geq 0$. In a

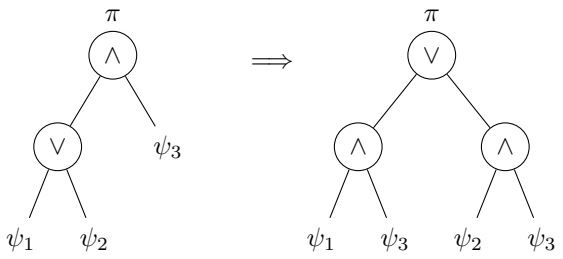



Figure 4.4: The equivalency and GCD preserving transformation used in the proof of Lemma 74 applied to a position $\pi$ in $\delta_{n-1}$. The subformula on the left is transformed to the subformula on the right. Obviously, both subformulas are equivalent and have the same GCD of all occurrences of a variable.

sense we are doing an "inverse" step of the step shown in Figure 4.2, constructing in a uniform way for each integer in $\delta$ a Tarski atomic formula.

A maximal set of pairwise conjunctively associated atomic positions $A$ has the property that any atomic position $\alpha \notin A$ is *not* conjunctively associated with at least one position from $A$. Next we prove a key lemma that uses Lemma 74 and asserts that each maximal pairwise conjunctively associated set of atomic positions in $\phi$ yields a pair in the result of `s-preproc-at`$(\phi, x)$:

**Lemma 75.** *Let $h \in \mathbb{N} \setminus \{0\}$. Let $\overline{\delta}$, $\delta$, and $\phi(b, c, x)$ be formulas of height $2h$ whose construction has been described above. Let $\{\alpha_1, \ldots, \alpha_n\}$ be a maximal set of pairwise conjunctively associated atomic positions in $\phi$. Then there exists a set $D = \left\{\left(\sqrt[d_1]{\hat{x}_1}, \alpha_1, \emptyset\right), \ldots, \left(\sqrt[d_n]{\hat{x}_n}, \alpha_n, \emptyset\right)\right\}$ of $n$ structural degree shift test points, where $d_i = p_{\alpha_i}$ for every $i \in \{1, \ldots, n\}$, such that the pair $(\phi_n, \hat{x}_n)$ is an element of the set of pairs $T$ returned by* `s-preproc-at`$(\phi, x)$. *Here we use the notation:*

$$\phi_0 = \phi[x \,/\, \hat{x}_0],$$
$$\phi_i = \Gamma(\phi_{i-1}, \alpha_i)[\hat{x}_{i-1} \,/\!/\, \sqrt[d_i]{\hat{x}_i}] \quad \text{for } i \in \{1, \ldots, n\}.$$

*Proof.* To prove the lemma it suffices to prove that $\left(\sqrt[d_j]{\hat{x}_j}, \alpha_j, \emptyset\right)$ is an element of the set $S$ computed by `s-shift-at` in the $(j-1)$-th level recursive invocation of `s-preproc-at` for every $j \in \{1, \ldots, n\}$, whereas the top-level invocation has level zero.

To begin with, observe that Lemma 73 ensures that for the top-level invocation of `s-preproc-at` ($j = 1$) we have: The set $S$ in line 4 of `s-preproc-at` is $\left\{\left(\sqrt[p_\alpha]{\hat{x}}, \alpha, \emptyset\right)\right\}_\alpha$, where $\alpha$ runs through all atomic positions of $\phi_0$. Therefore, in some iteration of the loop in step 4 the test point $\left(\sqrt[d_1]{\hat{x}_1}, \alpha_1, \emptyset\right)$ will be substituted into $\phi_0$, and the recursive call `s-preproc-at`$(\phi_1, \hat{x}_1)$ will follow.

Assume now that $j > 1$ and we are in the $(j-1)$-th level recursive call of `s-preproc-at` that was invoked on formula $\phi_{j-1}$ obtained by substituting the structural degree shift test points $\left\{\left(\sqrt[d_1]{\hat{x}_1}, \alpha_1, \emptyset\right), \ldots, \left(\sqrt[d_{j-1}]{\hat{x}_{j-1}}, \alpha_{j-1}, \emptyset\right)\right\}$ into $\phi_0$ one by one. We show that this recursive call computes a structural degree shift test point $\left(\sqrt[d_j]{\hat{x}}, \alpha_j, \emptyset\right)$, where $d_j = p_{\alpha_j}$, and uses it to make another $j$-th level recursive invocation of `s-preproc-at`. In the following we denote by $\delta_{j-1}$ the degree formula for $\phi_{j-1}$.

Observe now that the way we constructed the degree formulas $\delta_1, \ldots, \delta_{j-1}$ from $\delta_0$ can be mimicked by operations of Lemma 74: One step corresponds to one application of structural degree shift. Here notice that we use the fact that $p_{\alpha_1}, \ldots, p_{\alpha_{j-i}}$ are odd primes, so guard "$\hat{x} \geq 0$" is not introduced by a structural degree shift, and Lemma 74 can be used. Since $\alpha_j$ is conjunctively associated with each position in the set $\{\alpha_1, \ldots, \alpha_{j-1}\}$, $\alpha_j$ was not replaced with 0 in $\delta_j$. This ensures that it is a nonzero integer, because it was divided only by $p_{\alpha_1}$, $\ldots$, $p_{\alpha_{j-1}}$. Therefore, using Lemma 74 we obtain that $\gcd(\Gamma(\delta_{j-1}, \alpha_j)) = p_{\alpha_j}$, so the next invocation of `s-preproc-at` argument with $\phi_j$ will be definitely made.

Finally, observe that since $\{\alpha_1, \ldots, \alpha_n\}$ is a maximal set of pairwise conjunctively associated atomic positions in $\phi$ for the $n$-th level invocation we have: Each atomic position in $\delta_n$ is either zero or one. This ensures that in the recursive invocation of the level $n$ no structural degree shift w.r.t. an atomic position is possible, and `s-preproc-at` just returns $\{(\phi_n, \hat{x}_n)\}$. □

**Theorem 76.** *Let $h \in \mathbb{N} \setminus \{0\}$. Let $\overline{\delta}$, $\delta$, and $\phi(b, c, x)$ be formulas whose construction has been described above. Then algorithm* `s-preproc-at` *eventually constructs each member of the DNF of $\phi$.*

*Proof.* Let $C_1, \ldots, C_n$ be the DNF of $\phi$ returned by `the-dnf`$(\phi, ())$. First observe that each $C_j = \alpha_{j,1} \wedge \cdots \wedge \alpha_{j,m}$ consists of a set of $m$ atoms at pairwise conjunctively associated positions in $\phi$ (Theorem 55). Next we prove that $\alpha_{j,1}$, $\ldots$, $\alpha_{j,m}$ constitute a maximal set of pairwise conjunctively associated positions in $\phi$, i.e., for every atomic position $\alpha \notin \{\alpha_{j,1}, \ldots, \alpha_{j,m}\}$ there exists at least one element of $\{\alpha_{j,1}, \ldots, \alpha_{j,m}\}$ that is not conjunctively associated with $\alpha$.

Assume the opposite and let $\alpha$ be an atomic position that is conjunctively associated with all atomic formulas in $C_j$. This means that algorithm `the-dnf` appends $\alpha$—or potentially some other atomic formulas—to the conjunction $C_j$ during the recursive scanning of $\phi$. This means that $C_j$ is too short and as such does not appear in the output of `the-dnf`$(\phi, ())$; it could appear as a subformula of some longer conjunction though. This contradicts our assumption that $C_j$ is a disjunct of the DNF of $\phi$.

Since each $C_j = \alpha_{j,1} \wedge \cdots \wedge \alpha_{j,m}$ consists of a maximal set of pairwise conjunctively associated atomic positions, we can apply Lemma 75 to deduce that $C_j$ is contained in the set returned by `s-preproc-at`$(\phi, x)$. This finishes the proof of the theorem. □

To illustrate Lemma 75 and Theorem 76 on an example we run algorithm `s-preproc-at` on formula $\phi$ with degree formula showed in Figure 4.3. In Figure 4.5, Figure 4.6, and Figure 4.7 we sketch degree formulas obtained during execution by successive structural degree shifts.

Observe that `s-preproc-at` constructs possibly even more formulas than the DNF members, and that a DNF member is possibly constructed more than once by the algorithm. The reason for this is different order of applications of various structural degree shifts leading to the same resulting formula. By Proposition 52 we immediately obtain that the running time of `s-preproc-at` on $\phi$ of height $2h$—containing $2^{2h}$ distinct atomic formulas—is at least $2^{2^{h+1}-2}$.

From the practical point of view, our lower bound is not an issue at all: First, it is highly improbable that a formula of such a special form as $\phi$ occurs naturally in practice. Second, already for $h = 1$ every degree of an $x$-occurrence

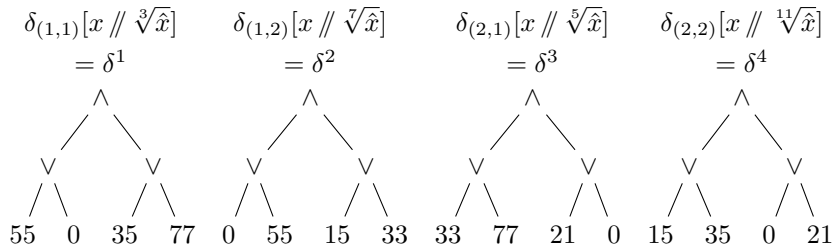



Figure 4.5: Degree formulas obtained from $\delta$ in Figure 4.3 by one structural degree shift. We condense w.r.t. positions $(1,1)$, $(1,2)$, $(2,1)$, and $(2,2)$, respectively.

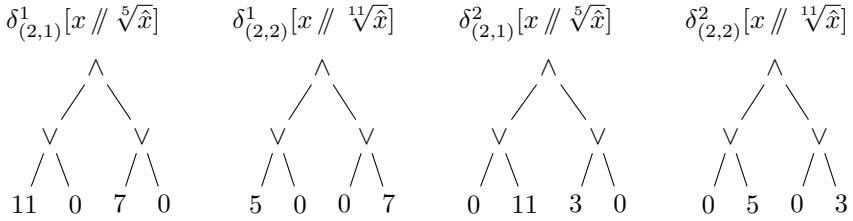



Figure 4.6: Degree formulas obtained from $\delta^1$ and $\delta^2$ in Figure 4.5 by one structural degree shift.

in $\phi$ is higher than hundred. As such this is behind the boundary of what is tractable at present by real quantifier elimination methods. After the DNF computation of $\phi$, however, the maximum degree of $x$ was 11, what is in our case tractable, e.g., by an efficient implementation of CAD but unfortunately not with a virtual substitution due to its degree restriction. Third, for higher values of $h$, where the DNF computation is really a bottleneck, the degrees are so high that no present method is able to cope with them. For example, for $h = 8$ the $2^{2h}$-th odd prime, i.e., the maximum degree of $\phi$, is 821647. Finally, one could simply bound the depth of recursion of `s-preproc-at` by a fixed constant to avoid exponential blowups.

### 4.2.2 Degree Shift Trying All Positions

It is not hard to see that the structural degree shift presented above is not really a generalization of the global degree shift of Section 4.1: It cannot even perform a global degree shift, because it tries a structural degree shift exclusively w.r.t. atomic positions. Here we improve on this and try to apply a degree shift w.r.t. any position in an input formula. Moreover, we also take advantage of incremental applications of the Marking technique. In this way we can replace already considered positions that yielded a structural degree shift test point with "false" and continue searching for other shift possibilities. The approach presented here is a generalization of the global degree shift, i.e., everything realizable by a global degree shift is realizable by our approach presented here.

Next we present algorithm `s-shift` that computes a set of structural degree shift test points by looking at all positions in an input formula when trying a

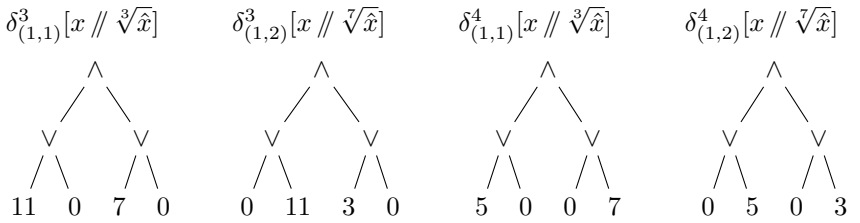



Figure 4.7: Degree formulas obtained from $\delta^3$ and $\delta^4$ in Figure 4.5 by one structural degree shift.

structural degree shift.

Algorithm `s-shift`$(\varphi, x)$.
Input: an $\wedge$-$\vee$-combination of Tarski atomic formulas $\varphi$, a variable $x$.
Output: a set $S$ of structural degree shift test points.

1. For each $\pi \in \mathtt{Pos}(\varphi)$ do
   1.1. Compute the GCD $d$ of all occurrences of $x$ in the formula $\varphi_\pi$, adjusting the degree of $x$ in atoms of the form $cx^k \varrho\, 0$ where necessary.
   1.2. Annotate position $\pi$ of $\varphi$ with $d$.
2. $S := \emptyset$
3. $R :=$ `get-indep`$(\varphi, ())$
4. Let $R$ be $\big\{(\pi_1, d_1), \ldots, (\pi_n, d_n)\big\}$. For $i := 1$ to $n$ do
   4.1. Add $\big(\sqrt[d_i]{x}, \pi_i, \{\pi_1, \ldots, \pi_{i-1}\}\big)$ to $S$.
5. Return $S$.

Algorithm `get-indep`$(\varphi, \pi)$.
Input: a Tarski formula $\varphi$ along with an integer annotation for every position in $\varphi$, a position $\pi \in \mathtt{Pos}(\varphi)$.
Output: a set $\big\{(\pi_1, d_1), \ldots, (\pi_n, d_n)\big\}$ such that positions $\pi_1, \ldots, \pi_n$ are pairwise independent and $d_i$ is the integer annotation of $\pi_i$ for every $i \in \{1, \ldots, n\}$.

1. If the annotation $d$ of $\pi$ is greater than 1, then return $\{(\pi, d)\}$.
2. If $\pi$ is a leaf, then return $\emptyset$.
3. For each child position $\pi|1, \ldots, \pi|k$ of $\pi$ do
   3.1. $R_i :=$ `get-indep`$(\varphi, \pi|i)$
4. Return $\bigcup_{i=1}^{k} R_i$.

Algorithm `s-shift` condenses $\varphi$ w.r.t. each position $\pi \in \mathtt{Pos}(\varphi)$ to determine where we could take advantage of a degree shift (step 1). Afterwards in step 3 an independent subset of the set of positions allowing a structural degree shift is computed. Finally, structural degree shift test points are constructed from this independent set of positions and returned in step 5.

Observe that $\emptyset$ is returned if and only if no structural degree shift is possible for $\varphi$. When a global degree shift is possible, then `s-shift` returns a set containing a single element $\big(\sqrt[d]{\hat{x}}, (), \emptyset\big)$ for some $d > 1$. For formula $\varphi$ from Example 70 `s-shift`$(\varphi, x)$ returns the same set of structural degree shift test points as `s-shift-at`, because a structural degree shift in $\varphi$ is possible only after condensing w.r.t. atomic positions.

Since `s-shift` computes the GCD of $\varphi_\pi$ for every position $\pi \in$ `Pos`$(\varphi)$, the algorithm needs to call a routine computing the GCD of a pair of integers at most $O\big(|\varphi| \log(|\varphi|)\big)$ times. Moreover, the subroutine `get-indep` runs in time $O(|\varphi|)$. To prove the correctness of `s-shift` we use Theorem 56 and Proposition 68:

**Theorem 77.** *Algorithms* `s-shift` *and* `get-indep` *meet their respective specifications. Let* $S = \big\{\big(\sqrt[d_1]{\hat{x}}, \pi_1, \mathcal{F}_1\big), \ldots, \big(\sqrt[d_n]{\hat{x}}, \pi_n, \mathcal{F}_n\big)\big\}$ *be the set returned by* `s-shift`$(\varphi, x)$. *Then* $\exists x(\varphi)$ *is equivalent to*

$$\exists \hat{x}\big(\gamma_1 \wedge \varphi_{\pi_1}^{\mathcal{F}_1}[x \mathbin{/\!\!/} \sqrt[d_1]{\hat{x}}]\big) \vee \cdots \vee \exists \hat{x}\big(\gamma_n \wedge \varphi_{\pi_n}^{\mathcal{F}_n}[x \mathbin{/\!\!/} \sqrt[d_n]{\hat{x}}]\big) \vee \exists x\big(\varphi^{\{\pi_1,\ldots,\pi_n\}}\big), \quad (4.4)$$

*where* $\gamma_i$ *is a guard of* $\big(\sqrt[d_i]{\hat{x}}, \pi_i, \mathcal{F}_i\big)$ *and we assume that in* $\varphi_{\pi_i}^{\mathcal{F}_i}$ *the same adjustments as in step 1.1 of* `s-shift` *were done for every* $i \in \{1, \ldots, n\}$.

*Proof.* To prove the theorem one proceeds similarly as in the proof of Theorem 71. The only difference is that instead of using Marking (Theorem 56) once we apply it here incrementally $n$ times to deduce that $\varphi$ is equivalent to the formula $\varphi_{\pi_1} \vee \varphi_{\pi_2}^{\{\pi_1\}} \vee \cdots \vee \varphi_{\pi_n}^{\{\pi_1,\ldots,\pi_{n-1}\}} \vee \varphi^{\{\pi_1,\ldots,\pi_n\}}$. □

Here we point to a rather subtle fact. The $n$ incremental applications of Marking in Theorem 77 suggest that an incremental computation of degree shift test points is correct: Whenever $\pi$ with GCD $d > 1$ of $\varphi_\pi$ is found in step 1.1, we produce a structural degree shift test point from $\pi$ and $d$, replace $\varphi$ with $\varphi^{\{\pi\}}$, and restart the loop. This could potentially lead to discovery of more shifting possibilities, because fewer atomic formulas would be used for computing the GCDs in step 1.1.

Integrating algorithm `s-shift` into `s-preproc-at` is straightforward: Just call `s-shift` instead of `s-shift-at` and use the returned degree shift test points. In the following we refer to this algorithm as `s-preproc`.

On the theoretical side we strongly believe that `s-preproc` still exhibits a "bad" behavior and computes the DNF of $\phi$, whose construction has been presented above. We even observed this behavior when running our prototype implementation of `s-preproc` on $\phi$ for $h \in \{1, 2, 3, 4, 5, 6\}$. Observe, however, that we cannot use the key Lemma 74 to model the behavior of `s-preproc`. The reason is that `s-preproc` replaces atomic formulas at positions $\{\pi_1, \ldots, \pi_{i-1}\}$ with "false" when substituting a degree shift test point $\big(\sqrt[d_i]{\hat{x}_i}, \pi_i, \{\pi_1, \ldots, \pi_{i-1}\}\big)$ into $\phi_{i-1}$. A formal proof of a lower bound result of `s-preproc` is therefore left for future work.

On the practical side, we expect `s-preproc` to perform much better than `s-preproc-at`, because it uses simpler formulas by using Marking incrementally. This prevents many symmetries, and possibly allows for more structural degree shifts. Similarly as with `s-preproc-at`, in the majority of practical applications one does not spot so high degrees of the variables in such a specific structure that a computation of the DNF would be triggered. Moreover, any degree lowering can be decisive for the applicability of virtual substitution to other quantified variables, so any improvement in comparison with the global degree shift counts.

## 4.3 Degree Shift and DNF

The main reason for using a degree shift heuristic is its potential to quickly reduce the maximum degree of a quantified variable in practice. In general, one is interested in lowering the degree of $x$ in $\varphi(\mathbf{u}, x)$ *without* raising the total number of variables, i.e., one would like to construct an equivalent $\varphi'(\mathbf{u}, \hat{x})$ with a strictly lower degree of $\hat{x}$ possibly without raising the degrees of $\mathbf{u}$. It seems that it is at least as hard as quantifier elimination to decide whether there exists such an equivalent $\varphi'$.

In this section we restrict ourselves to formulas that allow for lowering the maximum degree of $x$ in each member of their DNF, i.e., for every $C_i$ of the DNF $C_1 \vee \cdots \vee C_n$ we can lower the maximum degree of $x$ in $C_i$ by applying a global degree shift. Natural questions to ask are: Is it possible to detect efficiently that a formula has this property? Is it possible to take advantage of this property without computing the DNF of the formula?

The main result of this section is a lower bound result that states that one cannot hope for an improvement, and that the DNF computation is unavoidable in the worst-case. Similarly as in Section 4.2 we show how to construct for any $h \in \mathbb{N} \setminus \{0\}$ a Tarski formula $\phi$ such that the naive approach to compute the DNF of $\phi$ and to apply a degree shift in each DNF member is the best one can hope for. We begin our exposition with the following rather technical lemma:

**Lemma 78.** *Let $m$, $k \in \mathbb{N} \setminus \{0\}$ be such that $m > k$. Then there exists a set $A = \{a_1, \ldots, a_m\}$ of positive integers such that the following hold:*

*(i) For every subset $B = \{b_1, \ldots, b_k\} \subseteq A$ we have $\gcd(B) > 1$.*

*(ii) For every subset $C = \{c_1, \ldots, c_{k+1}\} \subseteq A$ we have $\gcd(C) = 1$.*

*Proof.* Let $l = \binom{m}{k}$ and let $p_1, \ldots, p_l$ be $l$ distinct primes. Let $g$ be arbitrary bijective mapping from the set of all $k$-element subsets of $\{1, \ldots, m\}$ into the set $\{p_1, \ldots, p_l\}$. For $i \in \{1, \ldots, m\}$ we define

$$a_i = \prod_{\substack{K \subseteq \{1, \ldots, m\} \\ |K| = k \\ i \in K}} g(K),$$

i.e., each $a_i$ is the product of those primes among $p_1, \ldots, p_l$ that are images of $k$-element subsets of $\{1, \ldots, m\}$ containing $i$. In the following we prove that (i) and (ii) hold for $A = \{a_1, \ldots, a_m\}$:

(i) Let $B = \{b_1, \ldots, b_k\} \subseteq A$. There exists a suitable permutation $\pi$ of the index set $\{1, \ldots, m\}$ such that $b_i = a_{\pi(i)}$ for every $i \in \{1, \ldots, k\}$. Let $p$ be the prime $g\big(\{\pi(1), \ldots, \pi(k)\}\big)$. The definition of $a_{\pi(i)}$ now ensures that $p$ divides $a_{\pi(i)}$ for every $i \in \{1, \ldots, k\}$, because $\pi(i) \in \{\pi(1), \ldots, \pi(k)\}$. Since $b_i = a_{\pi(i)}$ we deduce that $p$ divides $b_i$ for every $i \in \{1, \ldots, k\}$.

Moreover, there is no prime $q \neq p$ that divides $b_i$ for every $i \in \{1, \ldots, k\}$. Assume the opposite. Then $q = g(K)$ for some $k$-element subset of $\{1, \ldots, m\}$ such that $K \neq \{\pi(1), \ldots, \pi(k)\}$. Therefore, there exists $j \in \{1, \ldots, k\}$ such that $\pi(j) \notin K$. By the definition of $a_{\pi(j)}$ we obtain that $q \nmid b_j = a_{\pi(j)}$; a contradiction. This proves that $\gcd(B) = p$, so in particular (i) holds.

(ii) Let $C = \{c_1, \ldots, c_k, c_{k+1}\} \subseteq A$. Similarly as in the proof of (i), we deduce that there exists a suitable permutation $\pi$ of the index set $\{1, \ldots, m\}$ such that $c_i = a_{\pi(i)}$ for every $i \in \{1, \ldots, k\}$, and $\gcd\big(\{c_1, \ldots, c_k\}\big) = p$ for the prime $p = g\big(\{\pi(1), \ldots, \pi(k)\}\big)$. Observe that the prime $p$ does not divide $c_{k+1}$, because $\pi(k+1) \notin \{\pi(1), \ldots, \pi(k)\}$. This already shows that $\gcd(C) = 1$, i.e., (ii) holds. □

**Corollary 79.** *Let $m$, $k \in \mathbb{N} \setminus \{0\}$ be such that $m > k$. Then there exists a set $A = \{a_1, \ldots, a_m\}$ of positive odd integers such that the following hold:*

*(i) Let $j \in \{1, \ldots, k\}$. For any $B \subseteq A$ s.t. $|B| = j$ we have $\gcd(B) > 1$.*

*(ii) Let $j \in \{k+1, \ldots, m\}$. For any $C \subseteq A$ s.t. $|C| = j$ we have $\gcd(C) = 1$.*

*Proof.* It suffices to take $l = \binom{m}{k}$ odd primes in the proof of Lemma 78. The corollary then follows directly from the fact that the GCD of a set of integers divides the GCD of each of its subsets. □

Now we are ready to construct a Tarski formula $\phi(b, c, x)$ for every $h \in \mathbb{N} \setminus \{0\}$ such that it is "bad" in the following sense: First, it allows for a degree shift in each of its DNF disjuncts. Second, to take advantage of this property to lower the total degree of $x$ in $\phi$, one has to compute the whole DNF $C_1, \ldots, C_n$ of $\phi$.

The idea here is to construct $\phi(b, c, x)$ whose atomic formulas have $x$-degrees from the set $A$ from Corollary 79, whereas the number $m$ will be the number of all atomic formulas in the formula, and the number $k$ will be the number of atomic formulas occurring in every DNF disjunct.

Let $h$ be a positive integer. Corollary 79 guarantees that there exists a set $A = \{a_1, \ldots, a_m\}$ of $m = 2^{2h}$ odd natural numbers such that every $k$-element subset, $k = 2^h$, of $A$ has a nontrivial GCD, but every subset of $A$ with at least $k + 1$ elements has GCD 1. For every $i \in \{1, \ldots, m\}$ we define the atomic formula $\alpha_i$ of $x$-degree $a_i$ to be $bx^{a_i} + c \geq 0$. Here $b$ and $c$ are the parameters. We construct formula $\phi$ as a binary tree of height $2h$, which meets the following specification:

1. Every inner node has exactly two children.
2. An inner node with an odd distance from the root is an $\vee$-node.
3. An inner node with an even distance from the root is an $\wedge$-node.
4. The leaves of $\phi$ are pairwise distinct atomic formulas $\alpha_i$.

**Theorem 80.** *Let $h \in \mathbb{N} \setminus \{0\}$. Let $\phi(b, c, x)$ be the quantifier-free Tarski formula whose construction we have just described above. Then the following hold:*

*(i) Each member $C_i$ of the DNF of $\phi$ consists of exactly $2^h$ distinct atomic formulas of the formula $\phi$.*

*(ii) Each member $C_i$ of the DNF of $\phi$ allows for a global degree shift with some prime number $p$.*

*(iii) Let $\psi$ be an arbitrary $\wedge$-$\vee$-combination of any $l > 2^h$ distinct atomic formulas from $\phi$. Then no global degree shift for $x$ is applicable to $\psi$.*

*(iv) The formula $\exists x(\phi)$ is equivalent neither with "*true*" nor with "*false*."*

*Proof.* (i) Since $\phi$ meets specification 1–3 of Proposition 52, (i) follows directly from that proposition.

(ii) The construction of $\phi$ ensures that each member $C_i$ of the DNF of $\phi$ is a conjunction of the form $\bigwedge_j bx^{a_j} + c \geq 0$, where $j$ runs through some $2^h$-element subset of the set $\{1, \ldots, 2^{2h}\}$. Corollary 79 ensures that a global degree shift is applicable to the formula $\bigwedge_j bx^{a_j} + c \geq 0$, because the GCD of the degrees $a_j$ occurring in this formula is some odd prime number $p$. Therefore, Proposition 68 ensures that $\exists x(C_i)$ is equivalent to $\exists \hat{x}\big(C_i[x /\!\!/ \sqrt[p]{\hat{x}}]\big)$.

(iii) Observe that any $\wedge$-$\vee$-combination $\psi$ containing at least $2^h + 1$ distinct atomic formulas from $\phi$ contains formulas with at least $2^h + 1$ pairwise distinct powers of $x$. These powers are members of the set $A$, so Corollary 79 guarantees that their GCD is one, i.e., no global degree shift is possible for the formula $\psi$.

(iv) The construction of $\phi$ ensures that each member $C_i$ of the DNF of $\phi$ is a conjunction of the form $\bigwedge_j bx^{a_j} + c \geq 0$. Since every $a_j \in A$ is odd, it is easy to see that $\exists x(C_i)$ is equivalent to $b \neq 0 \vee c \geq 0$. Therefore, the whole formula $\exists x(\phi)$—that is equivalent to $\bigvee_i \exists x(C_i)$—is equivalent to $b \neq 0 \vee c \geq 0$ as well. □

We conclude this section by presenting two concepts that relate to DNF, shift, and detection of shifting possibilities in an input formula.

### Partial DNF

The *d-partial* DNF of a formula $\varphi$ is the formula obtained by computing the DNF of $\varphi$ using algorithm `the-dnf` while treating subformulas with depth $d$ as atomic formulas. Consequently, the disjuncts obtained from `the-dnf` are not necessarily conjunctions of atomic formulas anymore. After computing the $d$-partial DNF of $\varphi$ one can try a degree shift in each of the obtained disjuncts. This is made explicit in algorithm `s-preproc-d-DNF`:

Algorithm `s-preproc-d-DNF`$(\varphi, x, d)$.
Input: an $\wedge$-$\vee$-combination of Tarski atoms $\varphi$, a variable $x$, a $d \in \mathbb{N} \setminus \{0\}$.
Output: a finite set of pairs $\big\{(\varphi_i, x_i)\big\}$ such that $\varphi_i$ is an $\wedge$-$\vee$-combination of Tarski atoms, $x_i$ is a variable, and $\exists x(\varphi)$ is equivalent to $\bigvee_i \exists x_i(\varphi_i)$.

1. Compute the $d$-partial DNF $C_1, \ldots, C_n$ by calling `the-dnf`$(\varphi, ())$, while treating the subformulas of $\varphi$ with depth $d$ as atomic formulas.
2. $T := \emptyset$
3. For $i := 1$ to $n$ do
   3.1. Compute $T_i$ by calling `s-preproc`$(C_i, x)$ while bounding the recursion depth of `s-preproc` by one.
   3.2. $T := T \cup T_i$
4. Return $T$.

Notice that Theorem 80 implies also the following: The computation of the $d$-partial DNF of $\varphi$ in `s-preproc-d-DNF` cannot guarantee that one will be able to lower the degree of $x$ whenever each disjunct of the DNF of the formula $\varphi$ allows this.

### Conjunctive Associativity Degree Graph

As the final concept of this section we present the *conjunctive associativity degree graph* for $\varphi$. Its main potential is in detecting the possibly of degree shift in each DNF member of $\varphi$. It will turn out that a degree shift is possible in each DNF member of $\varphi$ if and only if every maximal clique of the conjunctive associativity degree graph for $\varphi$ has a nontrivial GCD. We first specify the properties of the graph and show how to compute it for $\varphi$:

Algorithm `conj-deg-graph`$(\varphi, x, \pi)$.
Input: an $\wedge$-$\vee$-combination of Tarski atoms $\varphi$, a variable $x$, a position $\pi \in$ `Pos`$(\varphi)$.
Output: a graph $(V, E)$, where $V$ is the set of atomic positions of $\pi(\varphi)$ containing $x$ and $(\pi_1, \pi_2) \in E$ if and only if $\pi_1$ is conjunctively associated with $\pi_2$ in $\pi(\varphi)$. Moreover, each $\pi \in V$ is annotated by one of the following: (1) a positive integer, (2) the symbol "$**$," (3) the symbol "$*0$," or (4) the symbol "$*1$."

1. If $\pi(\varphi)$ is an atomic formula not containing $x$, then return $(\emptyset, \emptyset)$.
2. If $\pi(\varphi)$ is an atomic formula containing $x$, then
   - 2.1. $V := \{\pi\}$
   - 2.2. If $\pi(\varphi)$ is an atomic formula of the form $cx^k \ \varrho \ 0$, $\varrho \in \{=, \neq\}$, then annotate $\pi$ with "$**$."
   - 2.3. If $\pi(\varphi)$ is an atomic formula of the form $cx^k \ \varrho \ 0$, $\varrho \in \{<, \leq, \geq, >\}$, then annotate $\pi$ with "$*l$," where $l \in \{0, 1\}$ and $l = k \mod 2$.
   - 2.4. Otherwise, annotate $\pi$ with the GCD of degrees of all $x$-occurrences in $\pi(\varphi)$.
   - 2.5. Return $(V, \emptyset)$.
3. For each child position $\pi|1, \ldots, \pi|n$ of $\pi$ do
   - 3.1. $(V_i, E_i) :=$ `conj-deg-graph`$(\varphi, x, \pi|i)$
4. If the top-level operator of $\pi(\varphi)$ is "$\wedge$," then
   - 4.1. Return $(V, E)$, where $V = \bigcup_{i=1}^{n} V_i$ and
   $$E = \bigcup_{i=1}^{n} E_i \cup \big\{(\pi_i, \pi_j) \in V_i \times V_j \mid i \neq j\big\}.$$
5. If the top-level operator of $\pi(\varphi)$ is "$\vee$," then
   - 5.1. Return $\big(\bigcup_{i=1}^{n} V_i, \bigcup_{i=1}^{n} E_i\big)$.

The algorithm is straightforward and it is easy to see that it is correct. Its running time depends on the size of the graph $(V, E)$, i.e., it is at most quadratic in the number of atomic formulas in $\varphi$. Note that the three starred annotations encode cases when an atomic formula allows for a degree adjustment before GCD computation with the semantic that any, even, or odd degree can be taken, respectively.

Notice that condensing $\varphi$ w.r.t. a position $\pi$ containing $x$ corresponds to deleting the vertices not connected with $\pi$. Similarly, a deletion of an atomic position $\pi$ corresponds to deletion of the vertex $\pi$.

Although similar, note that the conjunctive associativity degree graph for $\varphi$ computed by $\texttt{conj-deg-graph}\big(\varphi, x, ()\big)$ is different from the graph $G(\varphi, P)$ of Chapter 3, which we considered for a PC decomposition $P$ mainly for bound selection strategies.

**Proposition 81.** *Let* $(V, E) = \texttt{conj-deg-graph}\big(\varphi, x, ()\big)$. *Let* $C_1 \vee \cdots \vee C_n$ *be the DNF of* $\varphi$ *computed by* $\texttt{the-dnf}(\varphi, ())$. *Then there exists a member* $C_i = \alpha_1 \wedge \cdots \wedge \alpha_k$ *of the DNF of* $\varphi$ *containing* $x$ *if and only if there exists a maximal set of pairwise conjunctively associated atomic positions* $\pi_1, \ldots, \pi_l \in \texttt{Pos}(\varphi)$ *such that* $l \leq k$ *and* $\alpha_i = \pi_i(\varphi)$ *for every* $i \in \{1, \ldots, l\}$.

*Proof.* First notice that the $k - l$ formulas possibly contained in $C_i$ are those that do not contain $x$. The key idea of the proof has been presented already in the proof of Theorem 76, so we sketch here only the main points.

If $\pi_1, \ldots, \pi_l$ constitute a maximal pairwise conjunctively associated set of atomic positions, then from the execution of `the-dnf` one obtains the existence of a DNF member containing $\pi_1(\varphi), \ldots, \pi_l(\varphi)$.

For the converse, observe that whenever there exists a member $C_i = \alpha_1 \wedge \cdots \wedge \alpha_k$ there has to exist a maximal clique of $(V, E)$. If this was not the case, we could add another position $\alpha$ to the set $\alpha_1 \wedge \cdots \wedge \alpha_k$, and deduce that $C_i$ is not a member of the DNF at all; a contradiction. □

Observe that a maximal set of pairwise conjunctively associated atomic positions corresponds to a maximal clique of the conjunctive associativity degree graph $(V, E)$ by definition. Therefore, Proposition 81 guarantees that we can detect whether a degree shift in every member of the DNF of $\varphi$ is possible as follows: First, compute $(V, E) = \texttt{conj-deg-graph}\big(\varphi, x, ()\big)$. Then enumerate all maximal cliques of $(V, E)$ one by one, using the annotations of $V$ to compute for each clique whether its GCD is greater than one. For maximal clique enumeration one could use for example algorithms [11, 2].

## 4.4 Conclusions

In this chapter we have presented a technique used to lower the degree of a quantified variable in $\varphi$ called degree shift. First we have introduced the global degree shift and shown how to realize it by virtual substitution of Chapter 2 using shadow quantifiers. Then we have shown how to take advantage of the Boolean structure of $\varphi$ when shifting. It has turned out that this structural degree shift approach leads to the computation of the DNF of $\varphi$ in the worst-case. Afterwards, we have presented a generalization of the global degree shift that tries to apply a structural degree shift w.r.t. any position in $\varphi$. Furthermore,

we have shown that the following is an inherent problem when one wants to lower the maximum degree of a variable in $\varphi$ using shift: The computation of the DNF of $\varphi$ is unfortunately unavoidable in the worst-case. We have also discussed the $d$-partial DNF and its potential for yielding more degree shifts. Finally, we have introduced the conjunctive associativity degree graph for $\varphi$ for detecting whether a degree shift in each member of the DNF of $\varphi$ is possible.

Here we mention that the lower bound result presented in Section 4.3 is similar to the situation with bound selection in Chapter 3: We can try some heuristics, but to obtain the best possible degree shifting strategy we have to compute the DNF of an input formula. Doing this for every variable leads to an algorithm of non-elementary worst-case complexity. Nevertheless, in practice one should definitely try to take advantage of structural degree shift as much as possible, i.e., use the recursive approach trying all positions of Subsection 4.2.2 as a preprocessing step before `svs-scheme`. Even a partial lowering of the degree of a quantified variable in $\varphi$ could allow us to continue with virtual substitution for other quantified variables afterwards. To this end we mention that it would be interesting to have other sufficient conditions that could help us to quickly detect the possibility of structural degree shifts in practice. We leave derivation of such conditions for future work.

# Chapter 5

# Nonstandard and Standard Answers for Virtual Substitution

In this chapter we restrict ourselves to existential problems. Extended quantifier elimination generalizes the concept of regular quantifier elimination by providing in addition *answers*, which are descriptions of possible assignments for the quantified variables [82, 85]. Implementations of extended quantifier elimination [27, 29, 26] for the quadratic case via virtual term substitution [81, 83, 84, 51, 31] have been successfully applied to various problems in science and engineering [75, 74, 30, 67, 68, 69, 86, 72, 73, 71, 80, 36, 34, 35].

The answers produced by these implementations can include nonstandard symbols (infinitesimals and infinities), which are hard to interpret in practice. This has been explicitly criticized in the literature, e.g., in [20]. In our recent work [47] we have introduced a post-processing procedure to convert, for fixed values of parameters, all answers into standard real numbers when the degree of eliminated variables is at most two. We have furthermore demonstrated the successful application of an implementation of our method within the computer logic system Redlog [27] to a number of extended quantifier elimination problems from the scientific literature.

In addition to results presented elsewhere [47], we generalize here the post-processing procedure beyond the quadratic case. We adjust the procedure to our framework presented in the previous chapters.

We begin with Section 5.1 where we make ourselves familiar with the concept of extended quantifier elimination. In Section 5.2 we then present a concrete realization of this concept by virtual substitution and `vs-scheme`. For this we show how to apply our framework in presence of several quantifiers, and how to extract answers during the quantifier elimination process. Moreover, we discuss what information is provided by virtual substitution and the principal differences between our framework and quadratic extended quantifier elimination by virtual substitution considered in [47].

Section 5.3 is the technical core of this chapter; we describe and prove there the correctness of our post-processing procedure for removing nonstandard symbols from answers. We also show two example runs of the procedure and discuss

those example runs in detail. In Section 5.4 we make clear how to generalize our method to work in the structural setting of Chapter 3 and how to cope with possible degree shifts during quantifier elimination. Moreover, we discuss heuristics that can potentially improve the quality of obtained answers by finding parametric root descriptions representing rationals or even integers.

In Section 5.5, which is taken from our work [47], we revisit examples from the scientific literature where the application of quadratic extended quantifier elimination to various problems from planning, modeling, science, and engineering had yielded nonstandard answers. In all cases we can efficiently fix all nonstandard symbols to standard values using our implementation of the method for the quadratic case. This significantly improves the quality of the results from a practical point of view.

## 5.1 Extended Quantifier Elimination

For our purposes here, we restrict ourselves to existential problems

$$\vartheta(\mathbf{u}) = \exists x_n \ldots \exists x_1 \big(\varphi(\mathbf{u}, x_1, \ldots, x_n)\big),$$

where, as usual, $\mathbf{u} = (u_0, \ldots, u_{m-1})$ are the parameters, and $\varphi$ is an $\wedge$-$\vee$-combination of Tarski atomic formulas. We also agree that all right hand sides of the atomic formulas are zero, so the left hand sides of the atomic formulas are polynomials from $\mathbb{Z}[\mathbf{u}, x_1, \ldots, x_n]$.

*Extended quantifier elimination* applied to $\vartheta$ yields an *extended quantifier elimination result (EQR)* as follows:

$$\left[\begin{array}{c|ccc} \beta_1(\mathbf{u}) & x_1 = e_{1,1} & \ldots & x_n = e_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_k(\mathbf{u}) & x_1 = e_{k,1} & \ldots & x_n = e_{k,n} \end{array}\right].$$

The *conditions* $\beta_i(\mathbf{u})$ are quantifier-free Tarski formulas such that $\mathbb{R} \models \vartheta \longleftrightarrow \bigvee_{i=1}^{k} \beta_i$. In other words, $\bigvee_{i=1}^{k} \beta_i$ is a regular quantifier elimination result for $\vartheta$, and extended quantifier elimination generalizes regular quantifier elimination. The *answers* $\mathbf{e}_i = (e_{i,1}, \ldots, e_{i,n})$ are tuples of test points introduced in Chapter 2 meeting the following specification: Each $e_{i,j}$, where $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, n\}$, is of one of the following forms:

(a) a parametric root description $(f, S)$, where $f \in \mathbb{Z}[\mathbf{u}, x_{i+1}, \ldots, x_n][x_i]$,

(b) a parametric root description plus/minus a positive infinitesimal $(f, S)\pm\varepsilon$, where $(f, S)$ is as specified in (a),

(c) the nonstandard symbol $\pm\infty$.

For any parameter values $\mathbf{a} \in \mathbb{R}^m$, if $\vartheta(\mathbf{a})$ holds, then at least one $\beta_i(\mathbf{a})$ holds, and so does $\varphi(\mathbf{a}, \mathbf{e}_i\langle\mathbf{a}\rangle)$, where $\mathbf{e}_i\langle\mathbf{a}\rangle = \big(e_{i,1}\langle\mathbf{a}\rangle, \ldots, e_{i,n}\langle\mathbf{a}\rangle\big)$. We agree that "false" never occurs as a condition $\beta_i$. If $\vartheta$ itself is equivalent to "false," we possibly obtain the empty EQR [ ].

Here observe that the condition "$\varphi(\mathbf{a}, \mathbf{e}_i\langle\mathbf{a}\rangle)$ holds" in our definition of an EQR is quite vague, because the interpretation of $e_{i,j}\langle\mathbf{a}\rangle$ is not clear at all: What do $\infty\langle\mathbf{a}\rangle$ or $\big((f, S) - \varepsilon\big)\langle\mathbf{a}\rangle$ mean? In which structure should we interpret

these symbols? Is it possible that $e_{i,j}\langle\mathbf{a}\rangle$ is undefined when $e_{i,j} = (f, S)$? The main result of this chapter is that one can deal with something like this by avoiding answers of the forms (b) and (c) for fixed parameters, so that the obvious problem of interpreting something like $\infty\langle\mathbf{a}\rangle$ does not pop up at all. We will do this by replacing all answers of the forms (b) and (c) by computing answers of the form (a). Moreover, each $e_{i,j}\langle\mathbf{a}\rangle$ will be defined whenever $\beta_i$ holds by our construction. Before delving into this topic let us look at an example where things are straightforward and easy to interpret.

**Example 82.** Consider the input formula $\vartheta = \exists x \exists y(\varphi)$, where

$$\varphi(u, y, x) = uy + 3x^2 + 4x - u \leq 0 \wedge x - u \geq 0 \wedge u - y \geq 0.$$

A possible extended quantifier elimination result for $\vartheta$ is given by

$$\left[\begin{array}{l|ll} u \neq 0 \wedge 4u + 3 \geq 0 & y = \begin{pmatrix} y + 3u + 3, \\ (1, 1) \end{pmatrix} & x = \begin{pmatrix} x - u, \\ (1, 1) \end{pmatrix} \\ u \leq 0 \wedge \\ 3u^2 - 3u - 4 \leq 0 & y = \begin{pmatrix} y - u, \\ (1, 1) \end{pmatrix} & x = \begin{pmatrix} 3x^2 + 4x + u^2 - u, \\ \{(1, 1), (2, 1)\} \end{pmatrix} \end{array}\right].$$

From this extended quantifier elimination result we can derive a regular quantifier elimination result

$$(u \neq 0 \wedge 4u + 3 \geq 0) \vee (u \leq 0 \wedge 3u^2 - 3u - 4 \leq 0),$$

which can be simplified to $u \geq 0 \vee 3u^2 - 3u - 4 \leq 0$. Hence, $\vartheta$ holds if and only if the value $a \in \mathbb{R}$ of the parameter $u$ is greater or equal to $\alpha$, where $\alpha \approx -0.758306$ is the first real root of the polynomial $3u^2 - 3u - 4 \in \mathbb{Z}[u]$.

In the extended quantifier elimination result, the first row covers the case that $-0.75 \leq a$ and $a \neq 0$, while the second row covers $\alpha \leq a \leq 0$. Let us now consider some particular values $a$ of the parameter $u$:

- For $a = 2$, the condition $\beta_1$ in the first row holds, and the corresponding answers $\big(e_{1,1}\langle 2\rangle, e_{1,2}\langle 2\rangle\big)$ yield $y = -9$ and $x = 2$, because $-9$ is the first real root of $(y + 3u + 3)\langle 2\rangle$ and 2 is the first real root of $(x - u)\langle 2\rangle$. Indeed, we also have $\varphi\big(2, e_{1,1}\langle 2\rangle, e_{1,2}\langle 2\rangle\big)$. The condition $\beta_2$ in the second row, in contrast, does not hold for this parameter value. Considering the polynomial $\big(3x^2 + 4x + u^2 - u\big)\langle 2\rangle = 3x^2 + 4x + 2$, which is of real type (1), we see that $e_{2,2}\langle 2\rangle = \big(3x^2 + 4x + u^2 - u, \{(1, 1), (1, 2)\}\big)\langle 2\rangle$ is undefined. Therefore, $\varphi\big(2, e_{2,1}\langle 2\rangle, e_{2,2}\langle 2\rangle\big)$ does not make sense either.

- For $\alpha < a = -0.7525 < -0.75$, the condition $\beta_2$ in the second row holds, and the corresponding answers yield $y = -0.7525$ and $x = \frac{\sqrt{6997} - 800}{1200}$. Again, these three values satisfy $\varphi$, i.e., $\varphi\big(a, e_{2,1}\langle a\rangle, e_{2,2}\langle a\rangle\big)$. Now the condition in the first row does not hold. If we plug $a = -0.7525$ into the corresponding answers anyway, then we obtain $y = e_{1,1}\langle a\rangle = -0.7425$ and $x = e_{1,2}\langle a\rangle = -0.7525$, which do not satisfy $\varphi$.

- For $a = -0.5$, both conditions hold and yield two different sets of values satisfying $\varphi$, namely $y = e_{1,1}\langle a\rangle = -1.5$, $x = e_{1,2}\langle a\rangle - 0.5$ and $y = e_{2,1}\langle a\rangle = -0.5$, $x = e_{2,2}\langle a\rangle = \frac{\sqrt{7} - 4}{6}$, respectively.

- For $a = 0$ only the condition $\beta_2$ in the second row holds, but the answers in the first row happen to work as well. This shows that the conditions $\beta_i$ are sufficient but not necessary for the answers to be valid. ◇

Although we are focusing on the reals here, it is noteworthy that extended quantifier elimination is an established concept, which exists also for a variety of other important algebraic theories including the linear theory of valued fields [70], Presburger Arithmetic [49], initial Boolean algebras [63, 77], and certain term algebras [76].

## 5.2 Extended Quantifier Elimination by Virtual Substitution

Here we show how to realize extended quantifier elimination by our virtual substitution from Chapter 2. There we studied the elimination of a single existential quantifier. Therefore, we begin by introducing some notation and discussing what happens when one considers more existential quantifiers.

Given $\vartheta(\mathbf{u}) = \exists x\big(\varphi(\mathbf{u}, x)\big)$, we have shown in Chapter 2 how to compute a finite elimination set $E$ of test points $e$ such that

$$\exists x(\varphi) \longleftrightarrow \bigvee_{e \in E} \gamma_e \wedge \varphi[x /\!\!/ e]. \tag{5.1}$$

In the elimination set $E$ the $e$ are test points substituted for the quantified variable $x$ via our virtual substitution $[x /\!\!/ e]$. We have shown how to construct elimination sets so that each $e$ is of one of the following three forms: (1) a parametric root description $(f, S)$, (2) a parametric root description plus/minus a positive infinitesimal $(f, S) \pm \varepsilon$, or (3) the nonstandard symbol $\pm\infty$. We have shown how to realize our virtual substitution $[x /\!\!/ e]$ by `vs-at` for target atomic formulas and that $[x /\!\!/ e]$ naturally generalizes to arbitrary quantifier-free formulas. Each $\gamma_e$ in (5.1) is a guard of $e$ returned by $\texttt{guard}(e, x)$.

Equation (5.1) formally describes regular quantifier elimination of one quantifier $\exists x$ from $\vartheta$ that is implicitly realized by `vs-scheme`. For the elimination of several quantifiers, one assumes without loss of generality that the formula $\vartheta$ is prenex and processes the prenex quantifier block from the inside to the outside.

Using this formalism and building on our framework for virtual substitution, we now derive an extended quantifier elimination procedure for several existential quantifiers via virtual substitution. Given $\exists x_n \ldots \exists x_1\big(\varphi(\mathbf{u}, x_1, \ldots, x_n)\big)$, our intended result is a scheme

$$\left[\begin{array}{c|ccc} \beta_1(\mathbf{u}) & x_1 = e_{1,1} & \ldots & x_n = e_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_k(\mathbf{u}) & x_1 = e_{k,1} & \ldots & x_n = e_{k,n} \end{array}\right]$$

as defined in Section 5.1. We successively apply (5.1) to variables $x_1, \ldots, x_n$ using elimination sets $E_1, \ldots, E_n$, respectively, to obtain $\beta_1, \ldots, \beta_k$ as follows:

$$\bigvee_{e_n \in E_n} \cdots \bigvee_{e_1 \in E_1} \underbrace{\gamma_{e_n} \wedge \big(\cdots \wedge \big(\gamma_{e_1} \wedge \varphi[x_1 /\!\!/ e_1]\big) \cdots\big)[x_n /\!\!/ e_n]}_{\beta_i(\mathbf{u})}. \tag{5.2}$$

The index $i$ of $\beta_i$ describes one choice of test points $(e_1, \ldots, e_n)$ from the Cartesian product of elimination sets $E_1 \times \cdots \times E_n$. In practice, the $\beta_i$ obtained this way undergo sophisticated simplification methods such as those described in [28]. Recall from the previous section that $\beta_i$ which become "false" are ignored. Moreover, we assume that the elimination of a variable $x_j$ is done by a successful application of `vs-scheme`, while regarding all the remaining variables different from $x_j$ as parameters.

Looking carefully at `vs-scheme`, the fact that all except of a current to-be eliminated variable are regarded as parameters immediately implies that each $e_j$ occurring in $(e_1, \ldots, e_n) \in E_1 \times \cdots \times E_n$ is of one of the following forms:

(a) a parametric root description $(f_j, S_j)$, where $f_j \in \mathbb{Z}[\mathbf{u}, x_{j+1}, \ldots, x_n][x_j]$,

(b) a parametric root description plus/minus a positive infinitesimal $(f_j, S_j) \pm \varepsilon$, where $(f_j, S_j)$ is as specified in (a),

(c) the nonstandard symbol $\pm\infty$.

To obtain an EQR as defined in Section 5.1, it is therefore sufficient to merely collect those $\mathbf{e}_i \in E_1 \times \cdots \times E_n$ that yield $\beta_i$ different from "false."

Here we point to a difference between our EQR and EQR computed by means of the quadratic virtual substitution-based extended quantifier elimination [82, 85, 47]. To obtain an answer, the quadratic approach first exploits the quadratic virtual substitution in the same way as we do: It also remembers the elimination terms that are generated during regular quantifier elimination and pairs them with the quantifier-free formulas $\beta_i$ to obtain one row in an EQR. Afterwards, however, the quadratic approach is able to straightforwardly perform a *back-substitution* on the obtained elimination terms, because all of the elimination terms live in an extension $\mathfrak{L}'$ of the Tarski language $\mathfrak{L}$ containing in addition function symbols $\sqrt{}$ and $^{-1}$. The back-substitution therefore leads to a *diagonalized EQR* where each answer $e_{i,j}$ contains only the parameters. It is obvious that something like this cannot be generalized beyond degree four.

**Example 83.** Consider the formula $\vartheta = \exists x \exists y(\varphi)$, where $\varphi$ is the quantifier-free formula $y^2 - x + 1 = 0 \wedge x^2 - u = 0$. Using the quadratic virtual substitution method based on root expressions one obtains the following EQR for $\vartheta$:

$$\left[\; u - 1 \geq 0 \;\middle|\; y = \sqrt{x-1} \quad x = \sqrt{u} \;\right].$$

Carrying out the back-substitution on terms $\sqrt{x-1}$ and $\sqrt{u}$, one obtains the following diagonalized EQR:

$$\left[\; u - 1 \geq 0 \;\middle|\; y = \sqrt{\sqrt{u}-1} \quad x = \sqrt{u} \;\right],$$

where the answers for both $y$ and $x$ depend only on the parameter $u$. ◇

In our context a diagonalized EQR is an EQR such that an answer for a variable $x_j$ depends exclusively on the parameters $\mathbf{u}$, i.e., for any parametric root description $(f_j, S_j)$ in a diagonalized EQR we have $f_j \in \mathbb{Z}[\mathbf{u}][x_j]$ instead of $f_j \in \mathbb{Z}[\mathbf{u}, x_{j+1}, \ldots, x_n][x_j]$.

Since our framework does not use any extensions of the Tarski language $\mathfrak{L}$ to represent test points, it is not clear how to perform something similar to back-substitution on our test points. To eliminate a variable $x_{j+1}$ from an answer for

$x_j$, we essentially want to get a polynomial describing the common roots of the answers for $x_{j+1}$ and $x_j$. This suggests to use the resultant for diagonalizing an EQR. The following example shows that this does not always work.

**Example 84.** Consider the formula $\vartheta = \exists x \exists y\big(\varphi(u,v,y,x)\big)$, where $\varphi$ is

$$yx^2 + yv + 1 = 0 \wedge yx^2 + x + u = 0 \wedge 3x + u - 1 < 0.$$

One row of an EQR obtained for $\vartheta$ by our approach extracting answers from our virtual substitution scheme using (5.2) is:

$$\left[\ \beta \mid y = e_y \quad x = e_x\ \right].$$

The condition $\beta(u,v)$ is

$$\begin{aligned} &u^2 - 2u - 3v + 1 > 0 \wedge (v \neq 0 \vee u - 1 > 0) \wedge \\ &4u^4v - 12u^3v + 8u^2v^2 + 12u^2v + 20uv^2 - 4uv + 4v^3 - v^2 \leq 0, \end{aligned}$$

and the two test points $e_y$ and $e_x$ in the EQR are parametric root descriptions:

$$\begin{aligned} e_y &= \Big(f_y, \big\{(-1,1),(1,1)\big\}\Big), && \text{where } f_y = y(x^2+v)+1, \\ e_x &= \Big(f_x, \big\{(2,1),(3,1),(4,1)\big\}\Big), && \text{where } f_x = x^3 + x^2(u-1) + xv + uv. \end{aligned}$$

Since $f_y$ contains $x$, the EQR is obviously not diagonalized. To diagonalize this EQR, one has to eliminate $x$ from $f_y$ while keeping the properties of an EQR. A natural candidate for doing this is the resultant $r_y \in \mathbb{Z}[u,v][y]$,

$$r_y = u^2y - 2uvy^2 - 2uy + v^2y^3 + 2vy^2 + vy + y + 1$$

of $f_y$ and $f_x$ w.r.t. $x$. To compute a parametric root description $(r_y, S_y)$, one has to determine also root specifications $S_y$ that specify which of the roots of $r_y$ should be taken for concrete values of the parameters. In the following we prove that this is not possible by showing that for one choices of the parameters the root specification $(4,3)$ should be included in $S_y$ and for other choices it should not be included.

Fixing $u = 0$ and $v = -20$, the EQR specializes to:

$$\left[\ \text{true} \ \middle|\ y = \begin{pmatrix} y(x^2-20)+1, \\ \{(-1,1),(1,1)\} \end{pmatrix} \quad x = \begin{pmatrix} x^3 - x^2 - 20x, \\ \{(2,1),(3,1),(4,1)\} \end{pmatrix}\ \right],$$

which yields values $y = \frac{1}{4}$ and $x = -4$. Indeed, these four values satisfy $\varphi$. The resultant $r_y$ specializes for $u = 0$ and $v = -20$ to $400y^3 - 40y^2 - 19y + 1$. This polynomial is of real 3-type 4 and its third real root is $\frac{1}{4}$, which together with values for $u$, $v$, and $x$ satisfies $\varphi$ as we have shown above. This implies that the root specification $(4,3)$ should be definitely included in $S_y$.

On the other hand, for $u = 0$ and $v = \frac{3}{16}$ satisfying $\beta$ the specialized EQR yields values $y = -\frac{16}{3}$ and $x = 0$. Observe that the resultant $r_y$ specializes for $u = 0$ and $v = \frac{3}{16}$ to

$$\frac{9y^3 + 96y^2 + 304y + 256}{256}.$$

This univariate polynomial has three roots, namely $-\frac{16}{3} < -4 < -\frac{4}{3}$. Therefore, $(4, 3)$ should not be included in $S_y$ in this case; $(4, 1)$ should be included instead. Recall that it is not possible to include both root specifications in $S_y$. The reasons for this have been discussed below Theorem 25.

This shows that the computation of the resultant cannot be used to diagonalize an EQR. It seems that more information is needed to describe the values of $y$ in terms of the parameters. ◇

The structure of the answers in an EQR resembles the structure of formulas containing indexed root expressions as introduced by Brown [12] or cylindrical algebraic formulas considered by Strzeboński [64, 65]. However, as we have shown in Example 84, an answer polynomial $f_{x_{j+1}} \in \mathbb{Z}[\mathbf{u}, x_{j+2}, \ldots, x_n][x_{j+1}]$ over a region described as the zero set of another answer polynomial $f_{x_j} \in \mathbb{Z}[\mathbf{u}, x_{j+1}, \ldots, x_n][x_j]$ is not necessarily delineable.

Now we leave the diagonalization of EQRs, and discuss EQRs containing nonstandard symbols that are possibly introduced by `vs-scheme`. Looking at `vs-scheme` we discover that test points of the form $(f, S) \pm \varepsilon$ are taken for "EP," "SLB," and "SUB" candidate solutions. These candidate solutions themselves are returned by `at-cs` mostly for atomic formulas containing a strict relation "$\neq$," "$<$," or "$>$." The reason for this is that for such strict atomic formulas one needs a point from inside of its corresponding satisfying interval. This is what we ensure by our candidate solution sets introduced in Section 2.2.

To prevent the use of nonstandard expressions, Weispfenning [81] used in early versions of linear virtual substitution methods *arithmetic means* $\frac{1}{2}(z_i + z_j)$ for all pairs of terms $(z_i, z_j)$ generated by strict atomic formulas. However, the size of the elimination set then grows quadratically in the number of atomic formulas. Furthermore, one cannot apply any bound selection strategy. In experiments with early implementations this turned out to be critical for the practical performance of the method [17, 51]. For the quadratic case, there is an even more fundamental obstacle: Arithmetic means of two root expressions

$$\frac{1}{2}\left(\frac{a_1 \pm b_1\sqrt{c_1}}{d_1} + \frac{a_1 \pm b_2\sqrt{c_2}}{d_2}\right)$$

are themselves *not* root expression of the form $\frac{a \pm b\sqrt{c}}{d}$, so that Weispfenning's [84] virtual substitution rules for the quadratic case are not applicable at all. From a theoretical point of view, it easily follows from the existence of real quantifier elimination that suitable virtual substitutions for arithmetic means of root expressions exist. However, the practical applicability of this approach is quite questionable so that it has never been explicitly studied.

For our framework developed in thesis the above arguments apply as well: quadratic size of the elimination sets, impossibility to use bound selection strategies, arithmetic means of parametric root descriptions demanding special virtual substitution, and complications with structural elimination sets; all of these are reasons against avoiding the nonstandard symbols. Therefore, we simply use nonstandard symbols right from the start in all of our framework and show how to substitute them à la [84].

Let us return to our approach that possibly introduces nonstandard symbols. For the application of the elimination set as described in (5.1)—realized implicitly by `vs-scheme`—and thus for the computation of the $\beta_i$ with extended quantifier elimination, both $\pm\varepsilon$ and $\pm\infty$ are transparently translated into Tarski

formulas via virtual substitution of Section 2.3. At the same time, these symbols remain present in answers, as illustrated in the following example:

**Example 85.** On input of $\vartheta = \exists x \exists y\big(\varphi(u,y,x)\big)$, where $\varphi$ is

$$uy + 3x^2 < 0 \wedge x - y > 0 \wedge y - u > 0$$

we obtain an EQR containing nonstandard symbols:

$$\left[\ u < 0 \ \middle|\ y = \big(yu + 3x^2, (-1,1)\big) + \varepsilon_1 \quad x = \big(3x + u, (1,1)\big) - \varepsilon_2\ \right].$$

The condition $u < 0$ was obtained by successively substituting the answers $e_{1,1}$ and $e_{1,2}$ for $y$ and $x$, respectively.

Here we mention that the extended quantifier elimination by quadratic virtual substitution performs back-substitution of terms on the side of the answers to obtain a diagonalized EQR. During back-substitution nonstandard symbols cannot be removed but are propagated along the way. In the final result a single answer can even contain several of such nonstandard symbols. Using the quadratic extended quantifier elimination for the formula $\vartheta$ we obtain a diagonalized EQR by using back-substitution:

$$\left[\ u < 0 \ \middle|\ y = \frac{-3\varepsilon_1 - 3\varepsilon_2 - u}{3} \quad x = \frac{-3\varepsilon_1 - u}{3}\ \right].$$

To distinguish infinitesimals introduced at different stages of the elimination we indexed them accordingly. ◇

Given answers containing nonstandard symbols, it follows from the semantics of virtual substitution and from the correctness of `vs-scheme` that for any fixed values $\mathbf{a}$ of the parameters $\mathbf{u}$ satisfying a condition $\beta_i$ there exist real interpretations $\varepsilon_{i,j}\langle\mathbf{a}\rangle$ and $\infty_{i,j}\langle\mathbf{a}\rangle$ of all nonstandard symbols $\varepsilon_{i,j}$ and $\infty_{i,j}$ occurring in the $i$-th row of an EQR such that $\varphi\big(\mathbf{a}, \mathbf{e}_i\langle\mathbf{a}\rangle\big)$ whenever $\mathbf{a}$ satisfy $\beta_i$. Here we use the notation $\big((f_{i,j}, S_{i,j}) \pm \varepsilon_{i,j}\big)\langle\mathbf{a}\rangle = (f_{i,j}, S_{i,j})\langle\mathbf{a}\rangle \pm \varepsilon_{i,j}\langle\mathbf{a}\rangle$.

For Example 85 this means that for any value $a \in \mathbb{R}$ of the parameter $u$ satisfying the condition $u < 0$ there are positive real choices $\widetilde{\varepsilon_1} = \varepsilon_1\langle a\rangle$ and $\widetilde{\varepsilon_2} = \varepsilon_2\langle a\rangle$ so that the answers $(a, e_{1,1}\langle a\rangle + \widetilde{\varepsilon_1}, e_{1,2}\langle a\rangle - \widetilde{\varepsilon_2})$ satisfy $\varphi$.

It is noteworthy that the values for the nonstandard symbols have to be chosen differently in general, because they were introduced at different stages of the elimination, so semantically there is no connection between them. This will become clear in the next section where we constructively find an interpretation for one nonstandard symbol at a time.

Until recently users of extended quadratic quantifier elimination were left alone with nonstandard EQRs and the corresponding difficulties. Nevertheless, there has been a considerable record of applications of extended quadratic quantifier elimination in the literature. In [47] we have shown how to find standard answers for the quadratic case. In the next section we generalize that approach beyond the quadratic case to our framework in this thesis. Afterwards in Section 5.5 we are going to discuss some of the applications of our approach.

We conclude this section with the important observation that for unfixed parameters it is not possible in general to determine suitable real choices for nonstandard symbols:

**Proposition 86** (No Constant Standard Answers for Unfixed Parameters)**.** *Consider the formula* $\vartheta(u) = \exists x(u - x < 0 \wedge x - 1 < 0)$ *and the nonstandard extended quantifier elimination result*

$$\left[\ u - 1 < 0 \ \middle|\ x = \big(x - 1, (1,1)\big) - \varepsilon_1 \ \right].$$

*There is no standard choice* $\widetilde{\varepsilon_1} \in \mathbb{R}$ *such that*

$$\left[\ u - 1 < 0 \ \middle|\ x = \big((x + \widetilde{\varepsilon_1}) - 1, (1,1)\big) \ \right]$$

*is an extended quantifier elimination result for* $\vartheta$ *as well.*

*Proof.* Assume for a contradiction that $\widetilde{\varepsilon_1} \in \mathbb{R}$ is a suitable choice. Denote by $\widetilde{e}$ the parametric root description $\big((x + \widetilde{\varepsilon_1}) - 1, (1,1)\big)$. Since $\widetilde{e}$ does not contain the parameter $u$, we have $\widetilde{e}\langle a\rangle$ for any parameter value $a \in \mathbb{R}$.

By definition of extended quantifier elimination it follows for all values $a \in ]-\infty, 1[$ of the parameter $u$ that $(a, \widetilde{e}\langle a\rangle)$ satisfy $\varphi$, so in particular $\widetilde{e}\langle a\rangle < 1$. Since $\widetilde{e}\langle a\rangle = 1 - \widetilde{\varepsilon_1}$, we obtain $\widetilde{\varepsilon_1} > 0$. Therefore, for parameter value $b = 1 - \frac{\widetilde{\varepsilon_1}}{2}$ we have $b \in ]-\infty, 1[$, so $(b, \widetilde{e}\langle b\rangle)$ satisfy $\varphi$; in particular $b < \widetilde{e}\langle b\rangle$. Since $\widetilde{\varepsilon_1}$ is positive and $b = 1 - \frac{\widetilde{\varepsilon_1}}{2}$, we obtain $1 - \widetilde{\varepsilon_1} < b$. This means that $\widetilde{e}\langle b\rangle < b$, because $1 - \widetilde{\varepsilon_1} = \widetilde{e}\langle b\rangle$. Putting this all together we obtain $b < b$; a contradiction. □

## 5.3 Elimination of Nonstandard Symbols from Answers

Given an extended quantifier elimination result and prescribed values for all parameters, our goal is to compute answers containing only parametric root descriptions, i.e., containing no nonstandard symbols $\pm\varepsilon$ or $\pm\infty$. For instance, considering the formula from Example 85 and fixing the value $a = -2$ for the parameter $u$ we are going to obtain

$$\left[\ \text{true} \ \middle|\ y = \big(16y - 7, (1,1)\big) \quad x = \big(2x - 1, (1,1)\big) \ \right].$$

From the point-of-view of our method, it makes no difference whether the parameters are fixed after extended quantifier elimination or in advance. For the sake of a concise description, we are thus going to consider w.l.o.g. existential decision problems from now on. Recall that if the regular quantifier elimination result is "false," then the extended quantifier elimination result is [ ], i.e., empty. If the result is "true," then we assume for simplicity that the extended quantifier elimination result contains only one row, like

$$\left[\ \text{true} \ \middle|\ x_1 = e_{1,1} \quad \ldots \quad x_n = e_{1,n} \ \right]. \tag{5.3}$$

Our method is going to directly use an EQR containing the test points computed by our quantifier elimination framework to remove nonstandard symbols. It will turn out that the sole fact that the collected test points constitute an EQR gives us enough information to compute a standard EQR yielding standard values for the existentially quantified variables when the parameters are fixed. We begin with two preparatory lemmas on the semantics of virtual substitution.

**Lemma 87.** *Consider a quantifier-free Tarski formula $\varphi(x_1, \ldots, x_n)$. Assume that for each $i \in \{2, \ldots, n\}$ we have a parametric root description $\widetilde{e_i} = (f_i, S_i)$ such that $f_i \in \mathbb{Z}[x_{i+1}, \ldots, x_n][x_i]$. Assume furthermore that $\mathbb{R} \models \varphi \longrightarrow \gamma_i$ for a guard $\gamma_i$ of $(f_i, S_i)$. For each $i \in \{2, \ldots, n-1\}$ let $\alpha_i = \widetilde{e_i}\langle \alpha_{i+1}, \ldots, \alpha_n \rangle$, and let $\alpha_n = \widetilde{e_n}\langle\rangle$. Then*

$$\big\{\, \alpha \in \mathbb{R} \mid \mathbb{R} \models \varphi[x_2 /\!\!/ \widetilde{e_2}] \ldots [x_n /\!\!/ \widetilde{e_n}](\alpha) \,\big\} = \big\{\, \alpha \in \mathbb{R} \mid \mathbb{R} \models \varphi(\alpha, \alpha_2, \ldots, \alpha_n) \,\big\}.$$

*Proof.* The lemma follows directly from the semantics of virtual substitution formulated and proven correct in Theorem 25. □

**Lemma 88** (Commutation of Independent Virtual Substitutions)**.** *Consider a quantifier-free Tarski formula $\varphi(x_1, \ldots, x_n)$. Let $e_1 = (f_1, S_1)$, with $f_1 \in \mathbb{Z}[x_1]$ such that a guard of $(f_1, S_1)$ is equivalent to "*true*," i.e., $e_1\langle\rangle$ is defined. Furthermore, let $i \in \{2, \ldots, n\}$, let $e_i = (f_i, S_i)$ be such that $f_i \in \mathbb{Z}[x_{i+1}, \ldots, x_n][x_i]$, and let $\gamma_i(x_{i+1}, \ldots, x_n)$ be a quantifier-free formula that implies a guard of $e_i$. Then*

$$\mathbb{R} \models (\gamma_i \wedge \varphi)[x_i /\!\!/ e_i][x_1 /\!\!/ e_1] \longleftrightarrow (\gamma_i \wedge \varphi)[x_1 /\!\!/ e_1][x_i /\!\!/ e_i].$$

*Proof.* We need to prove that

$$\mathbb{R} \models (\gamma_i \wedge \varphi)[x_i /\!\!/ e_i][x_1 /\!\!/ e_1] \longleftrightarrow (\gamma_i \wedge \varphi)[x_1 /\!\!/ e_1][x_i /\!\!/ e_i]$$

holds for any values $a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in \mathbb{R}$ of $\{x_1, \ldots, x_n\} \setminus \{x_1, x_i\}$.

Assume first that $\mathbb{R} \models (\gamma_i \wedge \varphi)[x_i /\!\!/ e_i][x_1 /\!\!/ e_1](a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$. Theorem 25 (regarding $\{x_1, \ldots, x_n\} \setminus \{x_1\}$ as parameters) then guarantees that

$$\mathbb{R} \models (\gamma_i \wedge \varphi)[x_i /\!\!/ e_i]\big(e_1\langle\rangle, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n\big),$$

because a guard of $(f_1, S_1)$ is equivalent to "true," and $f_1 \in \mathbb{Z}[x_1]$. Since $\gamma_i$ contains only variables $x_{i+1}, \ldots, x_n$, we obtain that $(a_{i+1}, \ldots, a_n)$ satisfy $\gamma_i$, so from our assumption follows that a guard of $e_i$ holds as well. Therefore, using Theorem 25 (regarding $\{x_1, \ldots, x_n\} \setminus \{x_i\}$ as parameters) we deduce

$$\mathbb{R} \models (\gamma_i \wedge \varphi)\big(e_1\langle\rangle, a_2, \ldots, a_{i-1}, e_i\langle a_{i+1}, \ldots, a_n\rangle, a_{i+1}, \ldots, a_n\big).$$

Again, Theorem 25 (regarding $\{x_1, \ldots, x_n\} \setminus \{x_1\}$ as parameters) yields

$$\mathbb{R} \models (\gamma_i \wedge \varphi)[x_1 /\!\!/ e_1]\big(a_2, \ldots, a_{i-1}, e_i\langle a_{i+1}, \ldots, a_n\rangle, a_{i+1}, \ldots, a_n\big).$$

Finally, Theorem 25 (regarding $\{x_1, \ldots, x_n\} \setminus \{x_i\}$ as parameters) ensures that

$$\mathbb{R} \models (\gamma_i \wedge \varphi)[x_1 /\!\!/ e_1][x_i /\!\!/ e_i]\big(a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n\big).$$

The proof of the converse implication is similar, so we omit it. □

Now we are ready to prove the main result of this chapter:

**Theorem 89** (Computation of Standard Answers)**.** *Consider a closed Tarski formula $\vartheta = \exists x_n \ldots \exists x_1\big(\varphi(x_1, \ldots, x_n)\big)$. Assume that*

$$\left[\ \text{true} \;\middle|\; x_1 = e_1 \quad \ldots \quad x_n = e_n \ \right]$$

*is an EQR for $\vartheta$ such that each $e_i$ is of one of the following forms:*

(a) *a parametric root description* $(f_i, S_i)$*, where* $f_i \in \mathbb{Z}[x_{i+1}, \ldots, x_n][x_i]$*,*

(b) *a parametric root description plus/minus a positive infinitesimal* $(f_i, S_i) \pm \varepsilon$*, where* $(f_i, S_i)$ *is as specified in (a),*

(c) *the nonstandard symbol* $\pm\infty$*.*

*Then we can compute parametric root descriptions* $\widetilde{e_1}, \ldots, \widetilde{e_n}$ *meeting the specification (a) above such that the following is an EQR for* $\vartheta$ *as well:*

$$\left[\ \text{true} \ \middle|\ x_1 = \widetilde{e_1} \quad \ldots \quad x_n = \widetilde{e_n}\ \right].$$

*Proof.* For the sake of the proof, we are going to show that in addition to the required $\widetilde{e_1}, \ldots, \widetilde{e_n}$ and we can also compute real algebraic numbers $\alpha_1, \ldots, \alpha_n$ such that each $\alpha_i = \widetilde{e_i}\langle\alpha_{i+1}, \ldots, \alpha_n\rangle$. We represent these real algebraic numbers as pairs of univariate defining polynomials and open isolating intervals with rational endpoints. Throughout the whole prove we use the convention that $\gamma_i$ denotes a guard of the test point $e_i$ and $\widetilde{\gamma_i}$ denotes a guard of the test point $\widetilde{e_i}$ for any $i \in \{1, \ldots, n\}$.

Given $k \in \{1, \ldots, n\}$, it suffices to show that from

$$\left[\ \text{true} \ \middle|\ x_1 = e_1 \quad \ldots \quad x_k = e_k \quad x_{k+1} = \widetilde{e_{k+1}} \quad \ldots \quad x_n = \widetilde{e_n}\ \right]$$

and $\alpha_{k+1}, \ldots, \alpha_n$ we can compute suitable $\widetilde{e_k}$ and $\alpha_k$. Define

$$\begin{aligned} \varphi_k(x_k, \ldots, x_n) &= (\gamma_{k-1} \wedge \cdots \wedge \gamma_1 \wedge \varphi)[x_1 /\!\!/ e_1] \ldots [x_{k-1} /\!\!/ e_{k-1}], \\ \varphi_{k+1}(x_{k+1}, \ldots, x_n) &= (\gamma_k \wedge \varphi_k)[x_k /\!\!/ e_k], \end{aligned}$$

and observe that

$$\left[\ \text{true} \ \middle|\ x_k = e_k \quad x_{k+1} = \widetilde{e_{k+1}} \quad \ldots \quad x_n = \widetilde{e_n}\ \right] \tag{5.4}$$

and

$$\left[\ \text{true} \ \middle|\ x_{k+1} = \widetilde{e_{k+1}} \quad \ldots \quad x_n = \widetilde{e_n}\ \right] \tag{5.5}$$

are EQRs for $\exists x_n \ldots \exists x_k(\varphi_k)$ and $\exists x_n \ldots \exists x_{k+1}(\varphi_{k+1})$, respectively. On the basis of these definitions it is sufficient for our proof to compute suitable $\widetilde{e_k}$ and $\alpha_k$ such that

$$\left[\ \text{true} \ \middle|\ x_k = \widetilde{e_k} \quad x_{k+1} = \widetilde{e_{k+1}} \quad \ldots \quad x_n = \widetilde{e_n}\ \right]$$

is an EQR for $\exists x_n \ldots \exists x_k(\varphi_k)$ as well. We define furthermore

$$\begin{aligned} \xi(x_k, \ldots, x_n) &= \widetilde{\gamma_n} \wedge \cdots \wedge \widetilde{\gamma_{k+1}} \wedge \varphi_k, \\ \xi'(x_k) &= \xi[x_{k+1} /\!\!/ \widetilde{e_{k+1}}] \ldots [x_n /\!\!/ \widetilde{e_n}]. \end{aligned}$$

Lemma 87 applied to the quantifier-free formula $\xi(x_k, \ldots, x_n)$, the root expressions $\widetilde{e_{k+1}}, \ldots, \widetilde{e_n}$, and the real algebraic numbers $\alpha_{k+1}, \ldots, \alpha_n$ yields

$$\{\, \alpha \in \mathbb{R} \mid \mathbb{R} \models \xi'(\alpha) \,\} = \{\, \alpha \in \mathbb{R} \mid \mathbb{R} \models \xi(\alpha, \alpha_{k+1}, \ldots, \alpha_n) \,\}. \tag{5.6}$$

We distinguish three cases depending on the type of $e_k$:

(a) We have $e_k = (f_k, S_k)$, and $\gamma_k$ is a guard of $e_k$. We set $\widetilde{e_k} = e_k$ and $\widetilde{\gamma_k} = \gamma_k$. Since (5.5) is an EQR for the formula $\exists x_n \ldots \exists x_{k+1}(\varphi_{k+1})$ and the real algebraic numbers $\alpha_{k+1}, \ldots, \alpha_n$ correspond to $\widetilde{e_{k+1}}\langle\alpha_{k+2}, \ldots, \alpha_n\rangle$, $\ldots, \widetilde{e_n}\langle\rangle$, respectively, Theorem 25 ensures that $\mathbb{R} \models \varphi_{k+1}(\alpha_{k+1}, \ldots, \alpha_n)$. In particular $\mathbb{R} \models \gamma_k(\alpha_{k+1}, \ldots, \alpha_n)$. This means that $\widetilde{e_k}\langle\alpha_{k+1}, \ldots, \alpha_n\rangle$ is well-defined. This allows us to compute real algebraic number $\alpha_k = \widetilde{e_k}\langle\alpha_{k+1}, \ldots, \alpha_n\rangle$ from $\widetilde{e_k}$ and $\alpha_{k+1}, \ldots, \alpha_n$ as follows: First plug the real algebraic numbers $\alpha_{k+1}, \ldots, \alpha_n$ into $f_k$, obtaining $f\langle\alpha_{k+1}, \ldots, \alpha_n\rangle \in \mathbb{R}[x_k]$. This univariate polynomial is guaranteed to be of one particular real type $t$ specified by a root specification $(t, r) \in S_k$. Isolating all real roots, and evaluating the sign of $f\langle\alpha_{k+1}, \ldots, \alpha_n\rangle$ to the left, between, and to the right of the roots one finds the real type $t$ of the polynomial. Finally, extracting the $r$-th real root from the left yields $\alpha_k$.

(b) We have $e_k = (f_k, S_k) \pm \varepsilon$, and $\gamma_k$ is a guard of $e_k$. Since (5.4) is an EQR for formula $\exists x_n \ldots \exists x_k(\varphi_k)$, we have

$$\mathbb{R} \models \xi[x_k /\!\!/ (f_k, S_k) \pm \varepsilon][x_{k+1} /\!\!/ \widetilde{e_{k+1}}] \ldots [x_n /\!\!/ \widetilde{e_n}].$$

Using the fact that $\alpha_{k+1}, \ldots, \alpha_n$ are real algebraic numbers corresponding to $\widetilde{e_{k+1}}\langle\alpha_{k+2}, \ldots, \alpha_n\rangle, \ldots, \widetilde{e_n}\langle\rangle$, respectively, together with Theorem 25, we conclude that $\mathbb{R} \models \xi[x_k /\!\!/ (f_k, S_k) \pm \varepsilon](\alpha_{k+1}, \ldots, \alpha_n)$. Theorem 35(i) now guarantees the existence of some positive $\eta \in \mathbb{R}$ such that

$$\mathbb{R} \models \xi\big((f_k, S_k)\langle\alpha_{k+1}, \ldots, \alpha_n\rangle \pm \eta', \alpha_{k+1}, \ldots, \alpha_n\big)$$

for every real positive $\eta'$ strictly smaller than $\eta$. By (5.6) it follows that $\mathbb{R} \models \xi'\big((f_k, S_k)\langle\alpha_{k+1}, \ldots, \alpha_n\rangle \pm \eta'\big)$ for all real positive $\eta'$ strictly smaller than $\eta$.

Similarly as in case (a), we obtain from $(f_k, S_k)$ and $\alpha_{k+1}, \ldots, \alpha_n$ by real root isolation a real algebraic number $\alpha_k = (f_k, S_k)\langle\alpha_{k+1}, \ldots, \alpha_n\rangle$. Since $\xi'(\alpha_k \pm \eta')$ is guaranteed to hold for all positive real $\eta'$ strictly smaller than $\eta$, we are guaranteed to find after finitely many refinements of the isolating interval $\left]l, u\right[$ of $\alpha_k$ that $\mathbb{R} \models \xi'\big(l\big)$ when $e_k = (f_k, S_k) - \varepsilon$, or $\mathbb{R} \models \xi'\big(u\big)$ when $e_k = (f_k, S_k) + \varepsilon$.

When $e_k = (f_k, S_k) - \varepsilon$, we set $\widetilde{e_k}$ to be the parametric root description $\big(qx_k - p, (1, 1)\big)$, where $l = \frac{p}{q} \in \mathbb{Q}$, $q > 0$, so $qx_k - p \in \mathbb{Z}[x_k]$. When $e_k = (f_k, S_k) + \varepsilon$ we take $u = \frac{p}{q} \in \mathbb{Q}$, $q > 0$. In either case it is straightforward to construct a corresponding real algebraic number $\alpha_k = \widetilde{e_k}\langle\alpha_{k+1}, \ldots, \alpha_n\rangle$. Moreover, we set $\widetilde{\gamma_k} = \text{true}$ that is obviously a guard of $\widetilde{e_k}$. Then $\mathbb{R} \models (\widetilde{\gamma_k} \wedge \xi')[x_k /\!\!/ \widetilde{e_k}]$, and $n - k$ applications of Lemma 88 yield

$$\mathbb{R} \models (\widetilde{\gamma_n} \wedge \cdots \wedge \widetilde{\gamma_k} \wedge \varphi_k)[x_k /\!\!/ \widetilde{e_k}][x_{k+1} /\!\!/ \widetilde{e_{k+1}}] \ldots [x_n /\!\!/ \widetilde{e_n}],$$

i.e., we have an EQR for $\exists x_n \ldots \exists x_k(\varphi_k)$ such that $\widetilde{e_k}$ meets the specification (a) above.

(c) We have $e_k = \pm\infty$, and $\gamma_k$ is "true." Similarly to case (b), we observe that (5.4) is an EQR for $\exists x_n \ldots \exists x_k(\varphi_k)$. This implies

$$\mathbb{R} \models \xi[x_k /\!\!/ \pm\infty][x_{k+1} /\!\!/ \widetilde{e_{k+1}}] \ldots [x_n /\!\!/ \widetilde{e_n}].$$

Using the fact that $\alpha_{k+1}, \ldots, \alpha_n$ are real algebraic numbers corresponding to $\widetilde{e_{k+1}}\langle\alpha_{k+2}, \ldots, \alpha_n\rangle$, $\ldots$, $\widetilde{e_n}\langle\rangle$, respectively, and applying Theorem 25 we conclude that

$$\mathbb{R} \models \xi[x_k /\!\!/ \pm\infty](\alpha_{k+1}, \ldots, \alpha_n).$$

Theorem 35(ii) now guarantees that $\{\, \alpha \in \mathbb{R} \mid \mathbb{R} \models \xi(\alpha, \alpha_{k+1}, \ldots, \alpha_n) \,\}$ is unbounded from above when $e_k = \infty$ and from below when $e_k = -\infty$. Thus by (5.6) the set $\{\, \alpha \in \mathbb{R} \mid \mathbb{R} \models \xi'(\alpha) \,\}$ is unbounded from above/below as well. Using well-known bounds [3] on the roots of the univariate polynomials contained in $\xi'$, we compute a sufficiently large (or sufficiently small when $e_k = -\infty$) $\frac{p}{q} \in \mathbb{Q}$ such that $q > 0$ satisfies $\xi'$. We set $\widetilde{e_k}$ to be the parametric root description $\big(qx_k - p, (1,1)\big)$ and construct a corresponding real algebraic number $\alpha_k$. Furthermore, we set $\widetilde{\gamma_k} = \text{true}$ that is obviously a guard of $\widetilde{e_k}$. Exactly as in case (b), $\mathbb{R} \models (\widetilde{\gamma_k} \wedge \xi')[x_k /\!\!/ \widetilde{e_k}]$ together with $n-k$ applications of Lemma 88 yield

$$\mathbb{R} \models (\widetilde{\gamma_n} \wedge \cdots \wedge \widetilde{\gamma_k} \wedge \varphi_k)[x_k /\!\!/ \widetilde{e_k}][x_{k+1} /\!\!/ \widetilde{e_{k+1}}] \ldots [x_n /\!\!/ \widetilde{e_n}],$$

so in this case we also have an EQR for $\exists x_n \ldots \exists x_k(\varphi_k)$ such that $\widetilde{e_k}$ meets the specification (a) above. □

Notice that the constructive proof of Theorem 89 suggests to recompute the intermediate quantifier elimination results $\varphi_k$. In practice, there are arguments for saving these $\varphi_k$ during the extended quantifier elimination run. Consider, e.g., the following common optimization: Whenever some $\varphi_k$ heuristically simplifies to a disjunction $\varphi_{k,1} \vee \cdots \vee \varphi_{k,s}$, then the virtual substitution procedure would treat each $\varphi_{k,j}$ separately, i.e., like originating from several elimination set elements. In general, in the course of the application of Theorem 89 such transformations cannot be reconstructed exclusively from the EQR.

In the rest of this section we are going to illustrate Theorem 89 by means of two examples.

**Example 90.** We revisit Example 85 for the value $a = -2$ of the parameter $u$. For this choice, the formula $\varphi$ in $\vartheta = \exists x \exists y\big(\varphi(y,x)\big)$ specializes to

$$3x^2 - 2y < 0 \wedge x - y > 0 \wedge y + 2 > 0,$$

and our EQR containing nonstandard symbols becomes

$$\left[\ \text{true} \;\middle|\; y = \big(-2y + 3x^2, (-1,1)\big) + \varepsilon_1 \quad x = \big(3x - 2, (1,1)\big) - \varepsilon_2 \ \right].$$

Following the constructive proof of the theorem, we compute

$$\begin{aligned} \varphi_1(y,x) &= 3x^2 - 2y < 0 \wedge x - y > 0 \wedge y + 2 > 0, \\ \varphi_2(x) &= \text{true} \wedge \varphi_1\big[y /\!\!/ \big(-2y + 3x^2, (-1,1)\big) + \varepsilon\big] \\ &= \text{true} \wedge 3x^2 - 2x < 0. \end{aligned}$$

As in the proof, we proceed from the right to the left, i.e., our first step is fixing $x$ and computing a respective algebraic number $\alpha_x$. Since the answer for $x$ is $\big(3x - 2, (1,1)\big) - \varepsilon_2$, we are in case (b). Observe that

$$\left[\ \text{true} \;\middle|\; x = \big(3x - 2, (1,1)\big) - \varepsilon_2 \ \right]$$

is an EQR for $\exists x(\varphi_2)$. For $\xi = \text{true} \wedge \varphi_2$ we obtain that

$$\mathbb{R} \models \xi\big[x \mathbin{/\!\!/} \big(3x-2,(1,1)\big) - \varepsilon\big].$$

Theorem 35(i) guarantees that there exists a positive $\eta \in \mathbb{R}$ such that for all positive $\eta' \in \mathbb{R}$ strictly smaller than $\eta$ we have $\mathbb{R} \models \xi\Big(\big(3x-2,(1,1)\big)\langle\rangle - \eta'\Big)$. It follows that for all positive $\eta' \in \mathbb{R}$ strictly smaller than $\eta$ we also have $\mathbb{R} \models \xi'\Big(\big(3x-2,(1,1)\big)\langle\rangle - \eta'\Big)$, where $\xi' = \xi$ in this case.

To compute the real algebraic number $\alpha_x = \big(3x-2,(1,1)\big)\langle\rangle$ we first find out that the real 1-type of $3x-2$ is 1 with the only rational root $\frac{2}{3}$. The algebraic number $\alpha_x$ therefore represents $\frac{2}{3}$ as the root of $3x-2$ in an isolating interval $]-1,1[$.

Refining the isolating interval of $\alpha_x$ to $]\frac{1}{2},1[$ we find out that $\mathbb{R} \models \xi'(\frac{1}{2})$. Therefore we replace the nonstandard answer $x = \big(3x-2,(1,1)\big) - \varepsilon_2$ with the standard answer $x = \big(2x-1,(1,1)\big)$, which obviously represents the satisfying rational number $\frac{1}{2}$ found by interval refinement.

The second answer to consider is $y = \big(-2y+3x^2,(-1,1)\big) + \varepsilon_1$, which is again case (b). Now

$$\big[\ \text{true} \ \big|\ y = \big(-2y+3x^2,(-1,1)\big) + \varepsilon_1 \quad x = \big(2x-1,(1,1)\big)\ \big]$$

is an EQR for $\exists x \exists y(\varphi_1)$. Analogously to the previous step we obtain $\xi = \varphi_1$,

$$\begin{aligned} \xi' &= \xi\big[x \mathbin{/\!\!/} \big(2x-1,(1,1)\big)\big] \\ &= \text{true} \wedge -8y+3<0 \wedge -2y+1>0 \wedge y+2>0. \end{aligned}$$

To compute the real algebraic number $\alpha_y = \big(-2y+3x^2,(-1,1)\big)\langle\alpha_x\rangle$ we first determine that the real 1-type of $(-2y+3x^2)\langle\alpha_x\rangle$ is $-1$ with the root $\frac{3}{8}$, so $\alpha_y$ represents $\frac{3}{8}$ as the root of $(-2y+3x^2)\langle\alpha_x\rangle$ in an isolating interval $]-1,1[$.

Refining the isolating interval of $\alpha_x$ to $]0,\frac{7}{16}[$ we find that $\mathbb{R} \models \xi'(\frac{7}{16})$. We can therefore replace the nonstandard answer for $y$ with the standard answer $y = \big(16y-7,(1,1)\big)$, which obviously represents the satisfying rational $\frac{7}{16}$. Finally, the resulting standard EQR for $\vartheta$ looks as follows:

$$\big[\ \text{true} \ \big|\ y = \big(16y-7,(1,1)\big) \quad x = \big(2x-1,(1,1)\big)\ \big]. \quad \diamond$$

**Example 91.** Our second example addresses a rather straightforward but interesting generalization of extended quantifier elimination, which we have not discussed so far: Consider the valid sentence $\exists x \exists u(x-u<0)$. After elimination of $\exists u$ we essentially obtain $\exists x(\text{true})$. Our framework of Chapter 2 now computes an elimination set $\{\infty\}$ for $x$, so we obtain a corresponding answer $x = \infty$. In practice, however, we recognize that $x$ can be chosen arbitrarily, which implementations of extended quantifier elimination express using EQRs like:

$$\big[\ \text{true} \ \big|\ u = \infty \quad x = \text{arbitrary}_1\ \big].$$

For applying Theorem 89 such arbitrary variables have to be replaced with arbitrary but fixed answers that are interpretable in the context of the condition "true." It is easy to see that this is not a limitation of the theorem but a semantic consequence similar to the observation in Proposition 86.

Choosing, e.g., $\text{arbitrary}_1$ as $\big(x-2,(1,1)\big)$ and using Theorem 89 we compute

$$\begin{aligned}\varphi_1 &= x-u<0\\ \varphi_2 &= \text{true}\wedge\varphi_1[u/\!\!/\infty]\\ &= \text{true}\wedge\text{true}.\end{aligned}$$

For the answer $x=\big(x-2,(1,1)\big)$, we are in case (a), where we only have to compute a suitable $\alpha_x$ representing 2. Next, for $u=\infty$ we are in case (c) and obtain

$$\xi'=\text{true}\wedge 2-u<0.$$

Using, e.g., the Cauchy bound $1+\max\{\frac{2}{1}\}=3$ of $u-2$, we replace $u=\infty$ with the standard answer $u=\big(u-4,(1,1)\big)$, representing the integer 4, i.e., the Cauchy bound plus one. This yields the standard EQR

$$\big[\ \text{true}\ \big|\ u=\big(u-4,(1,1)\big)\quad x=\big(x-2,(1,1)\big)\ \big]\,.$$

Choosing, in contrast, $\text{arbitrary}_1$ as 3 we obtain the Cauchy bound $1+\max\{\frac{3}{1}\}=4$ and thus a standard EQR

$$\big[\ \text{true}\ \big|\ u=\big(u-5,(1,1)\big)\quad x=\big(x-3,(1,1)\big)\ \big]\,.\quad\diamond$$

## 5.4 Extensions of our Approach and Heuristics

Here we present extensions of our approach from the previous section. Furthermore, we also discuss heuristics for obtaining in our answers parametric root descriptions representing rational numbers or even integers.

Recall that in Chapter 4 we have shown how to realize global degree shift by virtual substitution. The idea was to model "$\sqrt[d]{\hat{x}}$" with a parametric root description $\big(x^d-\hat{x},((-1,0,1),1)\big)$ when $d$ is odd and with a parametric root description $\Big(x^d-\hat{x},\big\{((1,0,1),1),((1,0,-1,0,1),2)\big\}\Big)$ when $d$ is even. This modeling of a shift is extremely useful for our approach here: Assuming that an answer $x_k=\sqrt[d]{x_{k+1}}$ is a part of a processed EQR, we know that this is the case (a) in Theorem 89. Therefore, our construction obtains $e_k=\widetilde{e_k}$, computes a real algebraic number $\alpha_k$ using $e_k$ and $\alpha_{k+1},\ldots,\alpha_n$, and continues with processing the answer $x_{k-1}=e_{k-1}$. This means that our approach can be "extended" to handle global degree shift answers without any further adjustments.

Consider now the computation of $\widetilde{e_k}$ for some $e_k$. Recall that the ordering of the variables within the given EQR is such that quantifier elimination has taken place from the left to the right, while the construction of the standard answers proceeds from the right to the left. Here, the quantifier elimination direction has played an important role in the proof of Theorem 89: Although $e_{k+1},\ldots,e_n$ have been replaced with $\widetilde{e_{k+1}},\ldots,\widetilde{e_n}$, the answer $x_k=e_k$ is still valid. Taking that idea a bit further, we may replace $e_k$ with any valid expression without affecting the validity of either $e_1,\ldots,e_{k-1}$ or $\widetilde{e_{k+1}},\ldots,\widetilde{e_n}$.

In fact, it is sometimes possible to replace a parametric root description $e_k$ with a simpler one that represents a rational number or even an integer as follows: Before processing $e_k$, we check whether changing it to one of $e_k\pm\varepsilon$

yields a valid EQR for $\vartheta$ as well. This can be done by means of the virtual substitution

$$(\widetilde{\gamma_n} \wedge \cdots \wedge \widetilde{\gamma_{k+1}} \wedge \gamma_k \wedge \cdots \wedge \gamma_1 \wedge \varphi)$$
$$[x_1 /\!\!/ e_1] \ldots [x_{k-1} /\!\!/ e_{k-1}][x_k /\!\!/ e_k \pm \varepsilon][x_{k+1} /\!\!/ \widetilde{e_{k+1}}] \ldots [x_n /\!\!/ \widetilde{e_n}].$$

In the positive case, i.e., when the above ground formula is equivalent to "true," we process $e_k \pm \varepsilon$ instead of $e_k$. In terms of the proof of Theorem 89 this leads to the case (b), where we generally obtain parametric root description $\widetilde{e_k}$ representing a rational or even an integer; depending on whether a refined $]l, u[$ interval with integer endpoints satisfying $\xi'$ can be found.

Observe, however, that this heuristic does not work for a nontrivial degree shift $e_k$: A virtual substitution of $e_k \pm \varepsilon$ first expands every atomic formula by constructing a formula containing all the derivatives of the left hand side w.r.t. $x_k$ by using `expand-eps-at`. This means that a degree shift virtual substitution afterwards would not make sense because of degree divisibility restrictions we discussed in Section 4.1.

Notice that our approach is also compatible with the structural virtual substitution presented in Chapter 3: One can use `svs-scheme` in Section 5.2 to realize extended quantifier elimination. In this way one obtains answers of the form $x_k = (e_k, \pi_k, \mathcal{F}_k)$. Consequently, in the construction of Theorem 89 we have $\varphi_{k+1}(x_{k+1}, \ldots, x_n) = \big(\gamma_k \wedge (\varphi_k)^{\mathcal{F}_k}_{\pi_k}\big)[x_k /\!\!/ e_k]$ instead of $\varphi_{k+1}(x_{k+1}, \ldots, x_n) = (\gamma_k \wedge \varphi_k)[x_k /\!\!/ e_k]$. This is not a problem at all, and the proof remains valid, because $(\varphi_i)^{\mathcal{F}_i}_{\pi_i}$ implies $\varphi_i$ for every $i \in \{1, \ldots, n\}$. Therefore, using structural virtual substitution to realize extended quantifier elimination and our answer correction approach is much more an issue one has to deal with in implementation rather than in the formalism. To this end observe that the same arguments for saving intermediate structural quantifier elimination results we have discussed below the proof of Theorem 89 remain valid.

Looking carefully at the proof of Theorem 89, one observes that the computation of the real algebraic number $(f_k, S_k)\langle\alpha_{k+1}, \ldots, n\rangle$ involves computation with a polynomial with algebraic number coefficients. For this one can either use primitive elements to obtain $g_k \in \mathbb{Z}[x_k]$ such that $g_k(\alpha) = f_k\langle\alpha_{k+1}, \ldots, n\rangle(\alpha)\rangle$ for any $\alpha \in \mathbb{R}$. Another possibility is to use a recursive representation of real algebraic numbers [62] that works directly with $f_k\langle\alpha_{k+1}, \ldots, n\rangle(\alpha)$.

Finally, it is quite helpful in general to recognize rationals among all occurring real algebraic numbers. This holds in particular for the final $\alpha_1, \ldots, \alpha_n$, as they correspond to the values of $\widetilde{e_1}\langle\alpha_2, \ldots, \alpha_n\rangle, \ldots, \widetilde{e_n}\langle\rangle$, which may be roots of complicated high degree polynomials. For this one can use the following lemma.

**Lemma 92** (Rational Algebraic Numbers)**.** *Consider a real algebraic number $\alpha$ that is represented as the only root of $f = a_n x^n + \cdots + a_0$ in the interval $]l, u[$. Here $a_0, \ldots, a_n \in \mathbb{Z}$, $a_0 > 0$, $l, u \in \mathbb{Q}$, and $l > 0$. Assume furthermore that $\big]\frac{a_0}{u}, \frac{a_0}{l}\big[ \cap \mathbb{Z} = \{z\}$. Then $\alpha \in \mathbb{Q}$ if and only if $\alpha = \frac{a_0}{z}$.*

*Proof.* Let $\alpha \in \mathbb{Q}$. From $l > 0$ it follows that $\alpha > 0$, say $\alpha = \frac{p}{q}$, where $p, q \in \mathbb{Z}$, $p > 0$, $q > 0$. Using the Gaussian Lemma this admits the following factorization for suitable $b_0, \ldots, b_{n-1} \in \mathbb{Z}$:

$$\sum_{i=0}^{n} a_i x^i = (qx - p) \cdot \sum_{i=0}^{n-1} b_i x^i.$$

It follows that $a_0 = -b_0 p$, and we obtain $\alpha = \frac{-b_0 p}{-b_0 q} = \frac{a_0}{-b_0 q}$. On the other hand, $l < \frac{a_0}{-b_0 q} < u$, which is equivalent to $\frac{a_0}{u} < -b_0 q < \frac{a_0}{l}$, and it follows that $-b_0 q = z$. Together $\alpha = \frac{a_0}{-b_0 q} = \frac{a_0}{z}$. The converse implication is obvious. □

The lemma can be straightforwardly generalized to arbitrary intervals $]l, u[$.

## 5.5 Implementation and Application Examples

This section is taken from our work [47]. We include it here to illustrate the success of the post-processing method for the quadratic case. Note that the answers presented in this section are obtained by back-substitution on quadratic elimination terms, i.e., all EQRs here contain root expressions (with nonstandard symbols before our post-processing) rather than parametric root descriptions.

Recall that our framework generalizes and improves on the quadratic virtual substitution based on root expressions. An immediate consequence of this fact is that an implementation of our method from Section 5.3 would also be able to successfully process all of the examples presented here.

We have implemented our post-processing method for the quadratic case in Redlog, which is a part of the computer algebra system Reduce. Reduce is freely available under a modified BSD license.[1] Technically, our implementation is an extension of Redlog's original quadratic extended quantifier elimination `rlqea` by a switch `rlqestdans`, which toggles the computation of standard answers.

We are going to revisit a number of applications of extended quadratic quantifier elimination that have been documented in the scientific literature. In each case we are going to briefly explain the underlying problem, recompute the solutions with nonstandard answers, and finally compute solutions with standard answers using our approach as described in [47] and generalized in this chapter.

Since Redlog is very actively developed and improved, and the considered applications date back up to more than 15 years, the nonstandard answers obtained here partly differ from those reported in the literature. Of course, in such cases both variants are correct.

All computations have been carried out with the CSL variant of Reduce, revision 2465, using 4 GB RAM on a 2.4 GHz Intel Xeon E5-4640 running 64 bit Debian Linux 7.3.

### Computational Geometry

Besides many standard problems from computational geometry, the authors in [74] consider in Example 10 the reconstruction of a cuboid wireframe from a photography taken from the origin along the $x_3$-axis with a lens of focal length five.

The answers obtained by extended quantifier elimination is going to describe vectors $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3 \in \mathbb{R}^3$ generating the cuboid together with a vector $\mathbf{v} \in \mathbb{R}^3$ describing its translation from the origin. The input formula, which contains in addition points $\mathbf{i} \in \mathbb{R}^2$ on the camera sensor, contains 15 quantifiers:

$$\exists \mathbf{e}_1 \exists \mathbf{e}_2 \exists \mathbf{e}_3 \exists \mathbf{v} \forall \mathbf{i} \big( (\iota' \longleftrightarrow \pi_0) \wedge \exists k (59k\mathbf{v} = (100, 200, 295k + 295)) \big).$$

[1] `http://reduce-algebra.sourceforge.net`

The formula $\iota'(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{v}, \mathbf{i})$, which has been obtained by regular quantifier elimination earlier, generically describes that a point $\mathbf{i}$ lies in the image of a cuboid generated by $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, and translated by $\mathbf{v}$. The formula $\pi_0(\mathbf{i})$ is a quantifier-free description of one concrete image. The remaining part of the input formula fixes $\mathbf{i} = \left(\frac{100}{59}, \frac{200}{59}\right)$ to be the image of the origin of the cuboid.

Extended quantifier elimination yields "true" if and only if $\pi_0$ is a picture of a cuboid at all. In the positive case, the answers will provide suitable vectors $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, and $\mathbf{v}$.

For $\pi_0$ as considered in [74, Example 10], the extended quantifier elimination yields "true" together with the following nonstandard answers:

$$\mathbf{e}_1 = \left(5\infty_1, \frac{7\infty_1}{2}, \frac{5\infty_1}{2}\right), \qquad \mathbf{e}_2 = \left(\infty_1, 2\infty_1, \frac{-24\infty_1}{5}\right),$$

$$\mathbf{e}_3 = \left(\frac{-109\infty_1}{65}, \frac{53\infty_1}{26}, \frac{\infty_1}{2}\right), \qquad \mathbf{v} = \left(5\infty_1, 10\infty_1, \frac{59\infty_1 + 20}{4}\right).$$

Our method fixes $\infty_1 = 1$, which yields the following standard answers:

$$\mathbf{e}_1 = \left(5, \frac{7}{2}, \frac{5}{2}\right), \qquad \mathbf{e}_2 = \left(1, 2, -\frac{24}{5}\right),$$

$$\mathbf{e}_3 = \left(-\frac{109}{65}, \frac{53}{26}, \frac{1}{2}\right), \qquad \mathbf{v} = \left(5, 10, \frac{79}{4}\right).$$

The entire computation takes 189 s, of which the computation of the standard answers takes less than 1 ms.

## Motion Planning

Weispfenning [86] has studied motion planning problems in dimension two. Both the object to be moved and the free space between given obstacles are semilinear sets. Extended quantifier elimination is used to decide whether a geometrical object can be moved from an initial to a final destination in at most $n$ moves, where the trajectory of each move is a line segment. In the positive case, the answers describe the coordinates $\mathbf{u}_1, \ldots, \mathbf{u}_n \in \mathbb{R}^2$ of the object after each of the $n$ moves. Accordingly, the input formulas contain $2n$ variables in the prenex existential block.

We have applied our answer correction to three of the examples discussed in [86]. For the concrete input formulas and pictures of the scenery we refer to that publication.

For Example 6.4, we obtain the following nonstandard answers:

$$\mathbf{u}_1 = (5 - \varepsilon_1, 5 - \varepsilon_1), \quad \mathbf{u}_2 = \left(5 - \varepsilon_1, \frac{-2\varepsilon_1 + 1}{2}\right), \quad \mathbf{u}_3 = \left(9, \frac{9}{2}\right).$$

Our method fixes $\varepsilon_1 = \frac{3}{16}$, which yields the following standard answers:

$$\mathbf{u}_1 = \left(\frac{77}{16}, \frac{77}{16}\right), \quad \mathbf{u}_2 = \left(\frac{77}{16}, \frac{5}{16}\right), \quad \mathbf{u}_3 = \left(9, \frac{9}{2}\right).$$

The entire computation takes 60 ms, of which the computation of the standard answers takes less than 1 ms. Tables 5.1 and 5.2 summarize these results along with the two other examples.

| Example | $\mathbf{u}_1$ | $\mathbf{u}_2$ | $\mathbf{u}_3$ | Time |
|---|---|---|---|---|
| 6.4 | $(5-\varepsilon_1, 5-\varepsilon_1)$ | $\left(5-\varepsilon_1, \frac{-2\varepsilon_1+1}{2}\right)$ | $\left(9, \frac{9}{2}\right)$ | 0.06 s |
| 6.8 | $(0, 3+\varepsilon_1)$ | $(3-\varepsilon_1, 6)$ | $(7, 6)$ | 9.5 s |
| 6.9 | $(0, 3+\varepsilon_1)$ | $(3-\varepsilon_1, 6)$ | | 0.28 s |

Table 5.1: Summary of nonstandard answers and computation times for motion planning examples considered in [86].

| Example | $\varepsilon_1$ | $\mathbf{u}_1$ | $\mathbf{u}_2$ | $\mathbf{u}_3$ |
|---|---|---|---|---|
| 6.4 | $\frac{3}{16}$ | $\left(\frac{77}{16}, \frac{77}{16}\right)$ | $\left(\frac{77}{16}, \frac{5}{16}\right)$ | $\left(9, \frac{9}{2}\right)$ |
| 6.8 | 1 | $(0, 4)$ | $(2, 6)$ | $(7, 6)$ |
| 6.9 | 1 | $(0, 4)$ | $(2, 6)$ | |

Table 5.2: Summary of standard answers for motion planning examples considered in [86]. In all cases the time spent for the computation of $\varepsilon_1$ was less than 1 ms.

## Models of Genetic Circuits

Recently, symbolic methods for the identification of Hopf bifurcations in vector fields arising from biological networks or chemical reaction networks have received considerable attention in the literature [72, 73, 36, 80, 34]. Given a polynomial vector field, in [32] the authors introduced a method, which automatically generates first-order Tarski formulas describing the existence of a Hopf bifurcation in terms of the parameters. Then real quantifier elimination is applied to obtain corresponding necessary and sufficient conditions. For efficiency reasons, one often existentially quantifies all parameters and applies extended quantifier elimination. In the positive case, the answers provide one set of parameter values giving rise to a Hopf bifurcation.

Based on models introduced in [9], the publications [72, 73] used the approach sketched above to automatically derive the existence of Hopf bifurcations for the gene regulatory network controlling the circadian clock of a certain unicellular green alga. The input formula is

$$
\begin{aligned}
\exists(&0 < \mathbf{v}_1 \wedge 0 < \mathbf{v}_3 \wedge 0 < \mathbf{v}_2 \wedge 0 < \vartheta \wedge 0 < \gamma_0 \wedge 0 < \mu \wedge 0 < \delta \wedge 0 < \alpha \\
&\wedge \vartheta \cdot (\gamma_0 - \mathbf{v}_1 - \mathbf{v}_1\mathbf{v}_3^9) = 0 \wedge \lambda_1\mathbf{v}_1 + \gamma_0\mu - \mathbf{v}_2 = 0 \\
&\wedge 9\alpha(\gamma_0 - \mathbf{v}_1 - \mathbf{v}_1\mathbf{v}_3^9) + \delta(\mathbf{v}_2 - \mathbf{v}_3) = 0 \wedge 0 < \vartheta\delta + \vartheta\mathbf{v}_3^9\delta + 9\lambda_1\vartheta\mathbf{v}_1\mathbf{v}_3^8\delta \\
&\wedge 162\vartheta\mathbf{v}_3^{17}\alpha\mathbf{v}_1 + 162\vartheta\alpha\mathbf{v}_1\mathbf{v}_3^8 + 162\alpha\mathbf{v}_1\mathbf{v}_3^8\delta + \vartheta + 2\vartheta\mathbf{v}_3^9\delta + \vartheta^2\mathbf{v}_3^{18}\delta \\
&\quad + \vartheta\mathbf{v}_3^9 + 2\vartheta\delta + 81\alpha\mathbf{v}_1\mathbf{v}_3^8\vartheta\delta + 81\alpha\mathbf{v}_1\mathbf{v}_3^{17}\vartheta\delta + \delta^2 + \vartheta\delta^2 + \vartheta^2\delta + \vartheta^2 \\
&\quad + 2\vartheta^2\mathbf{v}_3^9 + \vartheta^2\mathbf{v}_3^{18} + 6561\alpha^2\mathbf{v}_1^2\mathbf{v}_3^{16} + 2\vartheta^2\mathbf{v}_3^9\delta + \delta + 81\alpha\mathbf{v}_1\mathbf{v}_3^8 + \vartheta\mathbf{v}_3^9\delta^2 \\
&\quad - 9\lambda_1\vartheta\mathbf{v}_1\mathbf{v}_3^8\delta = 0),
\end{aligned}
$$

for which we obtain the following nonstandard answers:

$$\gamma_0 = \frac{8\sqrt[9]{\infty_2}^{19}\infty_3+16\sqrt[9]{\infty_2}^{10}\infty_3+8\sqrt[9]{\infty_2}\infty_3}{729\infty_1^2\infty_2^3+1458\infty_1^2\infty_2^2+729\infty_1^2\infty_2-486\infty_1\infty_2^2\infty_3-486\infty_1\infty_2\infty_3+9\infty_2\infty_3^2},$$

$$\mu = \frac{729\infty_1^2\infty_2^3+1458\infty_1^2\infty_2^2+729\infty_1^2\infty_2-486\infty_1\infty_2^2\infty_3-486\infty_1\infty_2\infty_3+\infty_2\infty_3^2-8\infty_3^2}{8\infty_2^2\infty_3+16\infty_2\infty_3+8\infty_3},$$

$$\vartheta = \frac{\begin{array}{c}-6561\infty_1^4\infty_2^4-26244\infty_1^4\infty_2^3-39366\infty_1^4\infty_2^2-26244\infty_1^4\infty_2-6561\infty_1^4+4374\infty_1^3\infty_2^3\infty_3\\+13122\infty_1^3\infty_2^2\infty_3+13122\infty_1^3\infty_2\infty_3+4374\infty_1^3\infty_3-54\infty_1\infty_2\infty_3^3-54\infty_1\infty_3^3+\infty_3^4\end{array}}{\begin{array}{c}6561\infty_1^4\infty_2^5+32805\infty_1^4\infty_2^4+65610\infty_1^4\infty_2^3+65610\infty_1^4\infty_2^2+32805\infty_1^4\infty_2+6561\infty_1^4\\-8748\infty_1^3\infty_2^4\infty_3-34992\infty_1^3\infty_2^3\infty_3-52488\infty_1^3\infty_2^2\infty_3-34992\infty_1^3\infty_2\infty_3-8748\infty_1^3\infty_3\\+3078\infty_1^2\infty_2^3\infty_3^2+9234\infty_1^2\infty_2^2\infty_3^2+9234\infty_1^2\infty_2\infty_3^2+3078\infty_1^2\infty_3^2-108\infty_1\infty_2^2\infty_3^3\\-216\infty_1\infty_2\infty_3^3-108\infty_1\infty_3^3+\infty_2\infty_3^4+\infty_3^4\end{array}},$$

$$\mathbf{v}_1 = \frac{8\sqrt[9]{\infty_2}^{10}\infty_3+8\sqrt[9]{\infty_2}\infty_3}{729\infty_1^2\infty_2^3+1458\infty_1^2\infty_2^2+729\infty_1^2\infty_2-486\infty_1\infty_2^2\infty_3-486\infty_1\infty_2\infty_3+9\infty_2\infty_3^2},$$

$$\mathbf{v}_2 = \sqrt[9]{\infty_2}, \quad \mathbf{v}_3 = \sqrt[9]{\infty_2}, \quad \alpha = \infty_1, \quad \delta = 1, \quad \lambda_1 = \infty_3.$$

Our method fixes $\infty_1 = 1$, $\infty_2 = 9$, and $\infty_3 = 87481$, which yields the following standard solution:

$$\gamma_0 = \frac{69984800\cdot\sqrt[9]{9}}{616061191401}, \qquad \mu = \frac{3827162521}{69984800}, \qquad \vartheta = \frac{7652917261}{76056937210},$$

$$\mathbf{v}_1 = \frac{6998480\cdot\sqrt[9]{9}}{616061191401}, \qquad \mathbf{v}_2 = \sqrt[9]{9}, \qquad \mathbf{v}_3 = \sqrt[9]{9},$$

$$\alpha = 1, \qquad \delta = 1, \qquad \lambda_1 = 87481.$$

The entire computation takes 370 ms, of which the computation of the standard answers takes 140 ms.

## Mass Action Systems

We are now going to discuss another example about Hopf bifurcation. This time, the considered system is a chemical reaction system, viz. the famous and well-studied phosphofructokinase reaction. It has been firstly analyzed with symbolic methods in [37, Example 2.1]. We adopt here the first-order formulation discussed in [72, 73] following the approach sketched in the previous subsection.

We obtain nonstandard answers of the following form:

$$k_{21} = K_{21}(\infty_1,\infty_2,\infty_3,\infty_4,\varepsilon_1), \qquad k_{34} = \infty_1, \qquad k_{43} = \infty_2,$$
$$k_{46} = K_{46}(\infty_1,\infty_2,\infty_3,\infty_4,\infty_5,\varepsilon_1), \qquad k_{64} = \infty_5, \qquad k_{65} = \infty_3,$$
$$k_{56} = K_{56}(\infty_1,\infty_2,\infty_3,\infty_4,\infty_5,\varepsilon_1), \qquad \mathbf{v}_1 = \frac{\infty_2\infty_4}{\infty_1},$$
$$\mathbf{v}_2 = V_2(\infty_1,\infty_2,\infty_3,\infty_4,\infty_5,\varepsilon_1), \qquad \mathbf{v}_3 = \infty_4.$$

The nonstandard terms $K_{21}, \ldots, K_{56}, V_2$ are so large that we cannot explicitly display them here. To give an idea, $K_{46}$ would fill more than 16 pages in this document.

Our method fixes $\infty_1 = \infty_2 = \infty_3 = \infty_4 = 1$, $\infty_5 = 20$, and $\varepsilon_1 = 2(\sqrt{2}-1)$, which yields the following standard solution:

$$\begin{aligned}
&k_{21} = 3, && k_{34} = 1, && k_{43} = 1,\\
&k_{46} = \frac{\sqrt{3457}+1}{8}, && k_{64} = 20, && k_{65} = 1,\\
&k_{56} = \frac{-\sqrt{3457}+159}{6}, && \mathbf{v}_1 = 1,\\
&\mathbf{v}_2 = \frac{-\sqrt{3457}+159}{24}, && \mathbf{v}_3 = 1.
\end{aligned}$$

The entire computation takes 13.2 s, of which the computation of the standard answers takes 0.1 s.

## Sizing of Electrical Networks

Sturm [68, Section 5] has applied generic quantifier elimination to the sizing of a BJT amplifier. Description of a circuit is given as a set of operating point equations $E_1$ and a set of AC conditions $E_2$. For the concrete equations we refer to the mentioned publication. The system $E_1 \wedge E_2$ has to be solved w.r.t. the main variables $M = \{r_1, \ldots, r_8, c_3\}$ in terms of parameter variables $P = \{v_{\mathrm{cc}}, a_{\mathrm{high}}, a_{\mathrm{low}}, p, z_{\mathrm{in}}, z_{\mathrm{out}}\}$. Fixing values of the parameters to

$$v_{\mathrm{cc}} = 3,\ a_{\mathrm{high}} = 3,\ a_{\mathrm{low}} = 2,\ p = 12,\ z_{\mathrm{in}} = 5,\ z_{\mathrm{out}} = 5,$$

the answers contain one nonstandard term:

$$\begin{aligned}
&r_1 = \frac{4457058395}{5180672}, \quad r_2 = \frac{4457058395}{2590336}, && r_3 = -\frac{4457058395}{1295168},\\
&r_4 = -\frac{4182864929375836679}{128066211840000}, \quad r_5 = \infty_1, && r_6 = \frac{282999424999}{804520000},\\
&r_7 = 5, \quad r_8 = \frac{25509595605337086755}{20836792295619328}, && c_3 = \frac{647584}{13371175185}.
\end{aligned}$$

Our method fixes $\infty_1 = 1$, which yields a standard answer for $r_5$. The entire computation takes less than 2 ms, of which the computation of the standard answer takes less than 1 ms.

## A Linear Feasibility Example

In [45, Section 9] the authors have considered a small linear existential problem to demonstrate the difference between their conflict resolution method and the Fourier–Motzkin elimination method. The following are nonstandard answers for that problem computed by Redlog:

$$\begin{aligned}
&x_1 = -\frac{8}{13}, && x_2 = \frac{1-65\varepsilon_1}{65}, && x_3 = \frac{-14+13\varepsilon_2}{13},\\
&x_4 = \frac{-302-195\varepsilon_1+65\varepsilon_2}{130}, && x_5 = \frac{-30+26\varepsilon_2}{39}.
\end{aligned}$$

Our method fixes $\varepsilon_1 = \frac{1}{65}$ and $\varepsilon_2 = \frac{1}{13}$, which yields the following standard answers:

$$x_1 = -\frac{8}{13}, \quad x_2 = 0, \quad x_3 = -1, \quad x_4 = -\frac{30}{13}, \quad x_5 = -\frac{28}{39}.$$

The entire computation takes 3 ms, of which the computation of the standard answers takes less than 1 ms.

## 5.6 Conclusions

In this chapter we have introduced extended quantifier elimination as a general concept that in addition to a quantifier-free equivalent of an existentially quantified formula returns answers for the quantified variables. Then we have focused on our virtual substitution framework as one possible method for realization of extended QE. We analyzed the differences between extended QE realized within our framework and extended QE realized by quadratic virtual substitution. The main difference in this respect is the diagonalization of an EQR by means of back-substitution on terms, which cannot be carried out straightforwardly beyond degree two, and is impossible beyond degree four. Along the way we mentioned the similarity of our extended quantifier elimination results to cylindrical algebraic formulas.

The main result of this chapter was a generalization of the post-processing method—that converts nonstandard answers to standard answers for fixed values of the parameters—of [47] to work with our virtual substitution framework of Chapter 2. The main advantage of the generalization is that it is no longer limited to degree two. Then we argued that the generalization is compatible with the structural virtual substitution and degree shift presented in previous chapters. We have also discussed heuristics that can used to find in some cases simpler answers, i.e., parametric root descriptions representing rationals or even integers.

To demonstrate the success and usefulness of our generalized method, we have included in this chapter a section from our publication [47]. There we have revisited a number of applications of extended quantifier elimination by quadratic virtual substitution from the scientific literature, and computed standard answers by our implementation of the generalized method of this chapter for the quadratic case. Since our virtual substitution framework generalizes the quadratic virtual substitution approach, an implementation of the generalized method of this chapter together with the framework will be able to successfully process any of the examples considered in the section included from [47].

# Chapter 6

# Implementation and Computational Experiments

This chapter reports on our implementation of most of the concepts developed in this thesis. The main result presented here is a number of concepts and principles of our implementation of a virtual substitution-based real quantifier elimination, which are of general interest for future implementations as well. Our implementation is complete for degree three of a quantified variable. To the best of our knowledge, this is the first implementation of virtual substitution for the cubic case that is independent of other real quantifier elimination methods like CAD.

It is a key feature of our implementation that its data structures are extensible to cope with arbitrary but bounded degree of quantified variables. The implementation is modular and easily extensible beyond degree three: As we have seen in Chapters 2 and 3, working instantiations of `vs-scheme` and `svs-scheme` require implementations of the three procedures `at-cs`, `guard`, and `vs-prd-at`. This is the central principle of our implementation as well: Procedures for realizing the three algorithms constitute one independent submodule that can be easily provided with procedures for degree four and higher. At present we provide instantiations until degree three using the formula schemes of Appendix A.

Our implementation is a module of the computer logic system Redlog,[1] which itself is a part of the computer algebra system Reduce.[2] The whole system is open source and freely available under a very liberal Free-BSD license.

The most important part of Redlog we use is its powerful simplifier of quantifier-free Tarski formulas based on the techniques described in [28]. We also use Redlog's parser, and some standard formula manipulating procedures, e.g., for computing prenex or negation normal forms. Apart from these components, our implementation described here is independent of Redlog's original virtual substitution code for the quadratic case.

To demonstrate the efficiency of our implementation, we revisit around 200 real-world quantifier elimination problems from the sciences and engineer-

[1] `http://www.redlog.eu`

[2] `http://reduce-algebra.sourceforge.net`

ing. We compare the performance of the Redlog's original implementation of quadratic quantifier elimination with the implementation of our approach with and without clustering introduced in Chapter 2. The results of our computations are encouraging: Our implementation with clustering is generally able to eliminate at least as many quantifiers as the Redlog's original implementation. In a number of cases we can even eliminate more quantifiers. On the other hand, our implementation is on average 3.4 times slower than the Redlog's original implementation. The lengths of quantifier-free equivalents in comparison with quantifier-free equivalents obtained with the Redlog's original implementation are often very different: We observed cases when our implementation computed much shorter quantifier-free equivalents than the Redlog's original implementation and vice versa. Nevertheless, the results of our experiments are promising: Our research prototype implementation is able to compete with the Redlog's original implementation, which has been constantly being developed and improved for twenty years now.

One of the goals of this chapter is to give an impression of the design decisions and problems one faces when implementing a standalone virtual substitution-based quantifier elimination algorithm. We propose and implement data structures, heuristics, and approaches, which are of general interest with the hope that they will provide a basis for more efficient implementations.

The plan of this chapter is as follows: First we discuss how to implement a method for elimination of a single quantifier $\exists x$ from $\exists x(\varphi)$ based on `svs-scheme` developed earlier in this thesis. Then we show how to realize elimination of one block of existential quantifiers and discuss variable selection heuristics, which are of crucial practical importance here. Finally, we report on computational experiments conducted with our implementation and obtained results.

## 6.1 Elimination of One Existential Quantifier

As usual, we consider here the elimination of $\exists x$ from $\exists x\big(\varphi(\mathbf{u}, x)\big)$, where $\varphi$ is w.l.o.g. a quantifier-free $\wedge$-$\vee$-combination of Tarski atomic formulas. The right hand side of each atomic formula in $\varphi$ is zero.

The main building block of our implementation is the procedure `svs-scheme` for eliminating a single existential quantifier. Note, however, that we do not use `svs-scheme` as one monolithic procedure. We divided it into four parts that are used independently in our implementation:

1. degree shift,
2. prime constituent decomposition,
3. test points generation,
4. structural virtual substitution.

In the following we are going to describe each of these parts in detail.

### 6.1.1 Degree Shift and PC Decomposition

As the first step we always try to apply a degree shift on $\varphi$ and $x$. At present we support only the global degree shift technique of Section 4.1, though. If we

are successful, then we directly obtain an elimination set $\{\sqrt[d]{\hat{x}}\}$ for $\varphi$ and $x$. Otherwise, we continue by computing a prime constituent decomposition of the formula $\varphi$ w.r.t. $x$.

The procedure `PC-decomposition` is implemented almost exactly as we described it in Section 3.1: We first traverse the formula three times to find Gauss, co-Gauss, and atomic positions, computing candidate solutions along the way. After each of the first two traversals we temporarily replace found positions with "false" for optimization. In this way no atomic formula that is known to be a part of a Gauss or a co-Gauss prime constituent will be considered twice. Then we construct an independent set of positions using the strategy "bigger wins" discussed in Section 3.1. If we were able to compute a set of candidate solutions for each of the remaining positions, we generate and return a set of prime constituents accordingly. Otherwise, we return a FAILED message. This is the only place where we can fail, so later during test points generation and structural virtual substitution we rely on the fact that everything went well.

In the following we discuss a few points regarding the formula-traversing procedures `gaussposl`, `cogaussposl`, and `atposl` looking for Gauss, co-Gauss, and atomic positions of prime constituents in $\varphi$, respectively.

Each traversal procedure factorizes the left hand side of every atomic formula it looks at. For example, if the atomic formula $ax^3 - ax^2 + bx - b > 0$ is spotted, then it is considered in its factorized form, i.e., $(ax^2 + b)(x - 1) > 0$. According to the results of Subsection 2.2.1, candidate solutions for four atomic formulas $\left\{(ax^2 + b) \gtrless 0, (x - 1) \gtrless 0\right\}$ are computed afterwards. This leads to a candidate solution set containing parametric root descriptions of $ax^2 + b$ and $x - 1$.

We decided to always factorize the left hand side of an atomic formula, because we want to go as far with the virtual substitution method as possible: We do not want some higher degree polynomial that splits into factors of lower degrees to lead to a failure of the whole method.

In cases when a factor of a polynomial is of degree at most three and splits, one could consider ignoring the factorization of that factor. In our example we would view the left hand side $ax^3 - ax^2 + bx - b$ as is. This would yield candidate solutions containing exclusively parametric root descriptions of this third-degree polynomial. Observe that higher degree parametric root descriptions lead in general to longer and higher degree quantifier-free formulas after virtual substitution. For this reason we have not investigated this option further.

Recall from the description of `gaussposl` and `cogaussposl` in Section 3.1 that we need to decide whether the coefficients $c_i \in \mathbb{Z}[\mathbf{u}]$ of $f \in \mathbb{Z}[\mathbf{u}][x]$ in $f \varrho 0$, $\varrho \in \{=, \neq\}$, vanish. For this we do *not* use an expensive and complete real decision procedure. We rely on the efficient Redlog simplifier instead: If $\bigwedge_i c_i \neq 0$ simplifies to "false," then we know that $f \varrho 0$ is a Gauss or co-Gauss formula, respectively. If $f$ splits, then this simplification is done for each of its factors. Of course, this is an incomplete method, so we do not identify some Gauss or co-Gauss prime constituents in the worst case.

Let us now return to the process of generating candidate solutions during the formula traversals. Consider an atomic formula $ax^2 + bx + c > 0$, where $a$, $b$, $c \in \mathbb{Z}[\mathbf{u}]$. Recall than in algorithm `at-cs` of Section 2.5 we have to compute a set of candidate solutions for the atomic formula $bx + c > 0$ whenever $a\langle\mathbf{a}\rangle$ vanishes for some parameter values $\mathbf{a} \in \mathbb{R}^m$. This yields test points containing parametric root descriptions of $bx+c$ with a guard $b \gtrless 0$. It is absolutely correct to substitute these parametric root descriptions into the whole formula afterwards. Observe,

however, that such linear test points are really needed *only* for those $\mathbf{a} \in \mathbb{R}^m$ with $a\langle\mathbf{a}\rangle = 0$. For this reason we added a *relevance condition* to every test point in our implementation. The condition is by default "true." However, in a situation as mentioned above, we set the relevance conditions of the parametric root descriptions yielded by $bx+c > 0$ to formula $a = 0$. The relevance condition is used when substituting a test point containing a parametric root description: The condition is then simply conjunctively added to the guard of a test point. This can lead to simplifications of the resulting quantifier-free formula, or can create Gauss prime constituents in later elimination steps, because a relevance condition is always a conjunction of equations.

### 6.1.2 Test Points Generation and Substitution

After a successful PC decomposition, we are ready to compute a set of structural test points. For this we implemented a bound selection procedure and algorithms `PC-to-TPs` and `TPs-conflate` of Section 3.3 that are guaranteed to generate a structural elimination set for $\varphi$ and $x$ when given a valid PC decomposition $P$ of $\varphi$ w.r.t. $x$.

In our implementation we have only a very simple bound selection strategy: We simply count the number of upper and lower bound candidate solutions, and take the smaller set. Note, however, that the modular architecture allows us to easily extend our implementation with more sophisticated strategies like excluding the "EP" candidate solutions using Theorem 19 or the 0-1 ILP approach of Section 3.4. This is planned as a natural next step.

With a set of test points—obtained by bound selection and `PC-to-TPs`—in hand, we continue with conflation. As a conflation strategy we decided to "conflate everything what can be conflated." We made this explicit in `TPs-conflate`. The fact that this is not the only possible strategy along with the reasons for using it were discussed below Theorem 62.

After this the framework guarantees that we have a structural elimination set $E$ for $\varphi$ and $x$. We only need to substitute $E$ into $\varphi$ to obtain a quantifier-free equivalent of $\exists x(\varphi)$. The order in which one substitutes the test points into $\varphi$ does not play a role, and we do not do any special decision here either. Note, however, that one could first try "simpler" test points with the hope for getting "true." In such a case the remaining test points do not need to be considered at all. This would be of interest mainly with decision problems.

To substitute a test point $(e, \pi, \mathcal{F})$ into $\varphi$ we proceed as follows:

1. Construct a guard $\gamma_e$ of the test point using `guard`.
2. If the guard simplifies to "false," then return "false."
3. Otherwise continue by constructing formula $\varphi_\pi^{\mathcal{F}}$, and simplify it.
4. Then substitute $e$ into each atomic formula of the obtained formula by means of virtual substitution.

At this point we mention that step 1 is conceptually different from the Redlog's original implementation. A guard is there an inherent part of a test point (elimination term), so it is constructed already during test point generation. An advantage of our late guard construction is that only guards of those test points are constructed that will be really substituted. At the same time, this can be a

disadvantage for a bound selection strategy, when we take into account the irrelevant test points and make a decision that leads to a smaller set of test points, whereas a bigger set of test points could actually contain many irrelevant test points whose guards simplify to "false," i.e., they do not need to substituted at all—a better decision indeed.

To keep the Boolean structure of $\varphi$ untouched, we intentionally did not simplify $\varphi$ throughout the whole process of degree shift, PC decomposition, and test point generation. Otherwise, the whole framework of Chapter 3 would be inapplicable. In step 3 this is not the case anymore, and we can replace $\varphi_\pi^{\mathcal{F}}$ with any quantifier-free formula that is equivalent to it. Therefore, it is correct to use Redlog's simplifier of quantifier-free Tarski formulas at this point.

Notice that the substitution in step 4 could be done lazily taking again the Boolean structure of the target formula into account. Along with simplifications along the way, this could save some expensive virtual substitutions. In our implementation, however, we simply substitute $e$ into each atomic formula of the target formula and simplify the result afterwards.

Recall that during virtual substitution of a parametric root description $(f, S)$ into an atomic formula $g \varrho 0$, we first perform a pseudo division-like algorithm `pseudo-sgn-rem` of $g$ by $f$. As we can directly see from that algorithm, any assumptions leading to the positive or negative definiteness of $\operatorname{lc} f$ can help us to prevent the multiplication with $(\operatorname{lc} f)^2$. Since a guard often contains conditions like $\operatorname{lc} f \gtrless 0$, we use the guard along with the relevance condition for $(f, S)$ to determine whether $\operatorname{lc} f$ is positive or negative definite before calling `pseudo-sgn-rem`.

Finally, it is noteworthy that the static notion of "virtual substitution formula scheme" we use throughout the whole theoretical exposition in this thesis is imprecise from the implementation point of view. We implement virtual substitution by executable code that is given an atomic formula $g \varrho 0$ and a parametric root description $(f, S)$ and constructs $(g \varrho 0)[x /\!\!/ (f, S)]$. Since a virtual substitution formula scheme is obtained by combining virtual substitution formulas for lower degrees, this is easily achieved by recursively calling virtual substitution procedures for lower degrees.

## 6.2 Elimination of One Quantifier Block

Elimination of a block of existential quantifiers corresponds to a search through a *quantifier elimination tree*, which is a data structure storing formulas with partially eliminated quantifiers of an initial QE problem. A current state of the quantifier elimination process of the block $\exists x_n \dots \exists x_1$ from a formula $\exists x_n \dots \exists x_1\big(\varphi(\mathbf{u}, x_1, \dots, x_n)\big)$ is recorded by an instance of QE tree.

Each node of a QE tree contains a set of existentially quantified variables $Y = \{y_1, \dots, y_k\}$ to be eliminated and an $\wedge$-$\vee$-combination of Tarski atomic formulas $\phi$. Notice that each variable in $Y$ is either an original existentially quantified variable from $X$ or a shadow variable of some variable in $X$. The root of the tree contains $\big(\{x_1, \dots, x_n\}, \varphi\big)$. Children of a node $\big(Y, \phi\big)$ are obtained by structural virtual substitution of a structural elimination set $E$ for $\phi$ and $y \in Y$ into $\phi$. An edge of the tree represents structural virtual substitution of one particular structural test point $(e, \pi, \mathcal{F}) \in E$ for $y$ into $\phi$. If $(Y, \phi)$ is an inner node of a quantifier elimination tree, then $\exists Y(\phi) = \exists y_k \dots \exists y_1(\phi)$ is equivalent

to $\bigvee_i \exists Y'(\phi_i)$, where $\big\{(Y', \phi_i)\big\}_i$ are all the children of $(Y, \phi)$ and $Y' = Y \setminus \{y\}$ for some $y \in Y$. By induction it follows that for each internal node $(Y, \phi)$ we have $\exists Y(\phi) \longleftrightarrow \bigvee_j \exists Y_j(\psi_j)$, where $\big\{(Y_j, \psi_j)\big\}_j$ are all the leaves of the QE tree lying below $(Y, \phi)$.

Starting with a QE tree containing only the root node $\big(\{x_1, \ldots, x_n\}, \varphi\big)$, actual leaf nodes of the tree are systematically *expanded* until no quantified variable occurs in the leaves, or until a node cannot be expanded because of a failure. Node expansion of $(Y, \phi)$ is done by selecting a variable $y \in Y$ and computing a structural elimination set for $\phi$ and $y$.

An example QE tree of a successful elimination of the block $\exists z \exists y \exists x$ from a Tarski formula $\exists z \exists y \exists x \big(\varphi(u, v, x, y, z)\big)$ is shown in Figure 6.1. For space and readability reasons we show on each edge only a test point $e$, but in fact each edge stands for a structural virtual substitution of a structural test point $(e, \pi, \mathcal{F})$.

Since we consider only existential quantifiers, a common optimization is to split a disjunction and consider its members separately in the next elimination step. This was done for example in the node $(\{z\}, \varphi_{1,2})$, where we obviously computed a structural elimination set $\big\{(\infty, (), \emptyset)\big\}$, substituted $\infty$ into $\varphi_{1,2}$, and obtained a disjunction $\varphi_{1,2,1} \vee \varphi_{1,2,2}$.

Notice that a (structural) degree shift can create a path from the root to a leaf longer than $|X|$, where $X$ is the set of existentially quantified variables in the initial block. The reason for this is that an "elimination" of a variable by a structural degree shift elimination set introduces a shadow variable. This happens in our example in node $(\{y, z\}, \varphi_2)$, where we have obviously obtained a structural elimination set for $\varphi_2$ and $x$ consisting of $\big(\sqrt[3]{\hat{z}}, \pi_1, \emptyset\big)$ and $\big(\sqrt[2]{\hat{z}}, \pi_2, \{\pi_1\}\big)$.

Observe that the order of variables is not necessarily the same for paths from the root to two distinct leaves. As we can see, on the path from the root to $(\emptyset, \varphi_{2,1,1,1})$ we eliminated the variables in the order $x$, $z$, $\hat{z}$, $y$, which is obviously different from the order $x$, $z$, $y$, $\hat{z}$ on the path from the root to $(\emptyset, \varphi_{2,2,1})$.

Finally, in the context of extended quantifier elimination discussed in Section 5.2 we see that a path from the root to a leaf that is not "false" yields one line of an EQR for $\exists z \exists y \exists x \big(\varphi(u, v, x, y, z)\big)$.

### 6.2.1 Data Types

For the purposes of our implementation we do not model the edges of a QE tree explicitly. We instead include a virtual substitution $s$—that yields the node from its parent—in each node. Therefore, a QE node for our purposes is from now on a triple $(Y, \phi, s)$. The state when $s = \emptyset$ signalizes that $\phi$ is ready for next elimination step. If $s \neq \emptyset$, then $s$ needs to be applied to $\phi$ first in order to continue with the elimination of the next variable. In that case $s$ is of one of the following three forms:

(a) $y =$ arbitrary; This happens if and only if the variable $y$ does not occur in the parent of $\phi$.

(b) $y = \big(\sqrt[d]{\hat{y}}, \pi, \mathcal{F}\big)$; structural degree shift; Here $d$ is the GCD of all occurrences of $y$ in $\phi_\pi^{\mathcal{F}}$, and $\hat{y}$ is a shadow variable for $y$.

(c) $y = (e, \pi, \mathcal{F})$; Here $(e, \pi, \mathcal{F})$ is a structural test point produced during elimination set computation for the parent formula of $(Y, \phi, s)$ and the variable $y$.

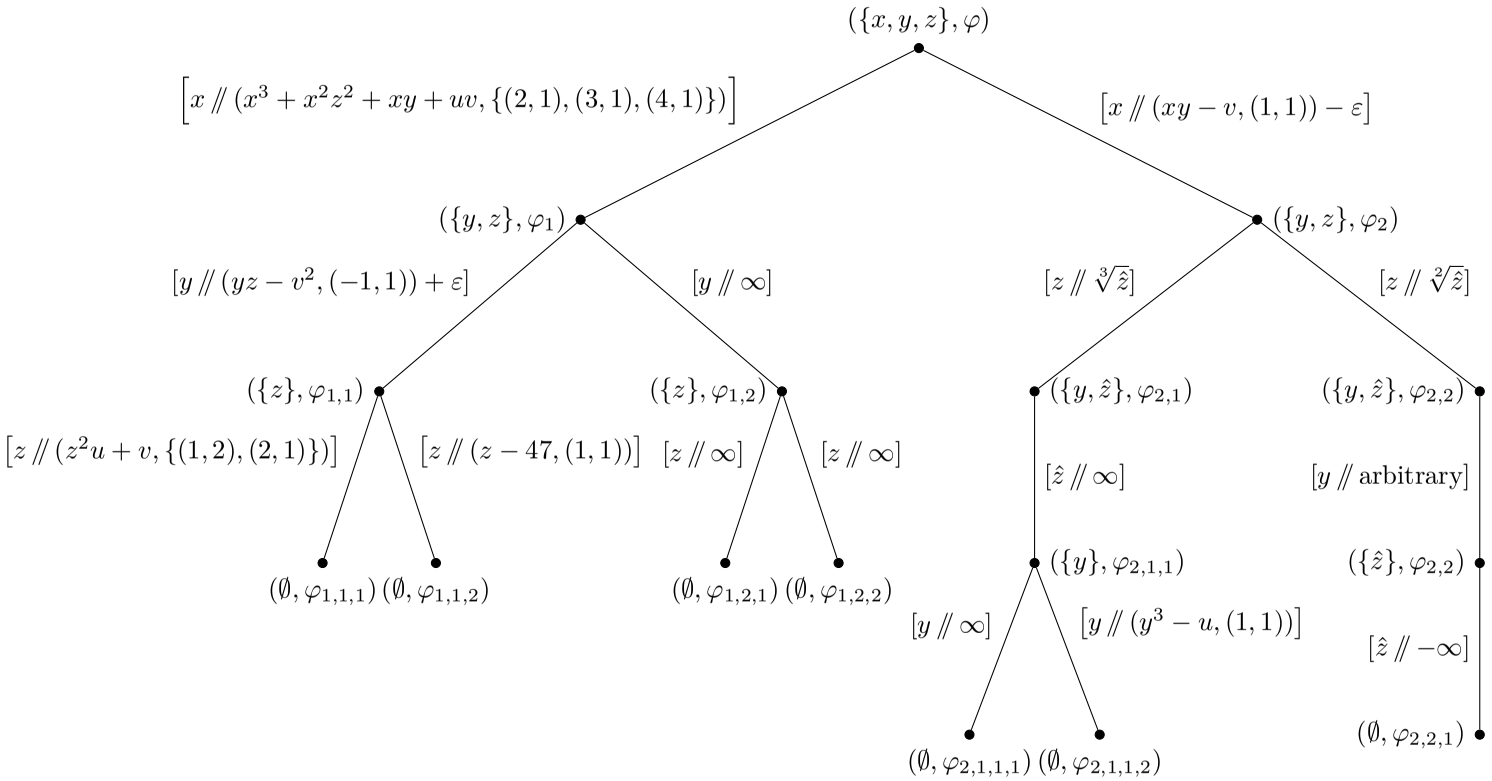



Figure 6.1: An example of a quantifier elimination tree showing the elimination of the quantifier block $\exists z \exists y \exists x$ from a formula $\exists z \exists y \exists x \big(\varphi(u,v,x,y,z)\big)$. Nodes of the tree represent quantifier elimination subproblems. Edges of the tree represent structural virtual substitutions. For each edge we for simplicity show only the test point $e$ of a structural test point $(e,\pi,\mathcal{F})$.

The nodes of a QE tree are divided into three disjoint categories containing *work*, *success*, and *failure* nodes, respectively. The work nodes are internal QE tree nodes that have not been expanded yet, success nodes are leaves that do not contain a variable for elimination, and failure nodes are nodes that cannot be properly expanded because of a failure. A *container* is a data structure containing all nodes of one category. Usually it is a queue or a stack. We will see that using a queue for the container of work nodes executes a breadth-first search through the QE tree, and a stack of work nodes executes a depth-first search through the QE tree. To prevent solving the same QE problem twice, we also have a hash table $H$ containing quantifier-free formulas. The hash table $H$ is populated during the search though a QE tree with formulas coming from the nodes of the tree. Keys of $H$ are pairs of the number of variables and the number of atomic formulas, and values of $H$ are quantifier-free formulas.

### 6.2.2 The Block Loop

The elimination of a block of quantifiers is done by `block-loop` given below. It manipulates with containers of QE tree nodes, expands work nodes, and calls elimination set computation and virtual substitution subroutines. In the following we denote the three containers of work, success, and failure nodes by $W$, $S$, and $F$, respectively.

Algorithm `block-loop`$(X, \varphi)$.
Input: a set of variables $X = \{x_1, \ldots, x_n\}$ and an $\wedge$-$\vee$-combination of Tarski atomic formulas $\varphi$.
Output: containers $S$ and $F$ containing QE nodes representing an equivalent of $\exists X(\varphi) = \exists x_n \ldots \exists x_1(\varphi)$. If $F = \emptyset$, then the elimination of $\exists X$ from $\exists X(\varphi)$ was successful, and $S$ contains a quantifier-free equivalent of $\exists X(\varphi)$.

1. $W := \emptyset$; $S := \emptyset$; $F := \emptyset$; $H := \emptyset$
2. Insert node $(X, \varphi, \emptyset)$ into $W$.
3. While $W \neq \emptyset$ do
   - 3.1. Take a node $(Y, \phi, s)$ from $W$.
   - 3.2. If $s \neq \emptyset$, then
     - 3.2.1. Apply the structural virtual substitution $s$ to $\phi$ and simplify.
     - 3.2.2. For each disjunction member $\psi$ of the resulting formula do
       - 3.2.2.1. Insert $(Y, \psi, \emptyset)$ into $W$.
     - 3.2.3. Continue with step 3.
   - 3.3. If $\phi$ is in the hash table $H$ and $s$ is not of the form $y =$ arbitrary, then continue with step 3.
   - 3.4. If $\phi$ is not in the hash table $H$, then insert $\phi$ into $H$.
   - 3.5. If $\phi$ is "true," then
     - 3.5.1. $W := \emptyset$; $S := \emptyset$; $F := \emptyset$
   - 3.6. If $Y = \emptyset$, then
     - 3.6.1. Insert $(Y, \phi, \emptyset)$ into $S$.
     - 3.6.2. Continue with step 3.

3.7. `expand-node`$(Y, \phi, \emptyset)$

4. Return $S$ and $F$.

Algorithm `block-loop` begins by inserting the root node of the to-be constructed QE tree into the container of working nodes $W$. Each iteration of the block loop in step 3 first checks whether we need to apply $s$ to the taken working node. If this is the case, we carry out the structural virtual substitution $s$ accordingly using the approach described in Subsection 6.1.2. Putting in step 3.2.2 all the resulting disjuncts into the working container $W$, we ensure that they will be taken later with $s = \emptyset$.

In step 3.3 we have $s = \emptyset$, i.e., no structural virtual substitution needs to be applied to $\phi$, and we can continue with the next variable. First we check whether $\phi$ was already considered as a QE subproblem. If this is not the case, we add it to the hash table $H$. If there is a variable to eliminate, then we expand the node $(Y, \phi, \emptyset)$ in step 3.7. Otherwise, we are done with the node and insert it into the success container. If we found "true," then we empty all three containers. This means that we will eventually add "true" to the success container after fixing substitutions $y =$ arbitrary for every remaining $y$ in $Y$. This becomes clear from `expand-node` that constructs the children of an internal node in the QE tree:

Algorithm `expand-node`$(Y, \phi, s)$.
Input: a node $(Y, \phi, s)$ of a QE tree with $s = \emptyset$.
Output: no meaningful return value; We directly modify $W$, $S$, and $F$.

1. For each variable $y \in Y$, considered in some prescribed ordering, do
   1.1. If $y$ does not occur in $\phi$, then
      1.1.1. Insert $(Y \setminus \{y\}, \phi, y = \text{arbitrary})$ into $W$.
      1.1.2. Return.
2. For each variable $y \in Y$, considered in some prescribed ordering, do
   2.1. $E_y :=$ `s-shift`$(\phi, y)$
   2.2. If $E_y \neq \emptyset$, then
      2.2.1. For each $(\sqrt[d]{\hat{y}}, \pi, \mathcal{F}) \in E_y$ do
         2.2.1.1. Insert $\big((Y \setminus \{y\}) \cup \{\hat{y}\}, \phi, y = (\sqrt[d]{\hat{y}}, \pi, \mathcal{F})\big)$ into $W$.
      2.2.2. Denote by $\mathcal{P}$ the positions of all structural degree shift test points in $E_y$.
      2.2.3. Insert $\big(Y, \phi^{\mathcal{P}}, \emptyset\big)$ into $W$.
      2.2.4. Return.
3. If variable selection strategy 1, then
   3.1. Let $y$ be the minimum variable in $Y$ w.r.t. some prescribed ordering.
   3.2. Compute a structural elimination set $E_y$ for $\phi$ and $y$.
   3.3. If this failed, then
      3.3.1. Insert $(Y, \phi, \emptyset)$ into $F$.
      3.3.2. Return.
   3.4. For each $(e, \pi, \mathcal{F}) \in E_y$ do

3.4.1. Insert $\big(Y \setminus \{y\}, \phi, y = (e, \pi, \mathcal{F})\big)$ into $W$.

3.5. Return.

4. If variable selection strategy 2, then

4.1. For each variable $y \in Y$, considered in some prescribed ordering, do

4.1.1. Compute a structural elimination set $E_y$ for $\phi$ and $y$.

4.1.2. If this was successful, then

4.1.2.1. For each $(e, \pi, \mathcal{F}) \in E_y$ do
Insert $\big(Y \setminus \{y\}, \phi, y = (e, \pi, \mathcal{F})\big)$ into $W$.

4.1.2.2. Return.

4.2. Insert $(Y, \phi, \emptyset)$ into $F$.

4.3. Return.

5. $M := \emptyset$

6. For each variable $y \in Y$ do

6.1. Compute a structural elimination set $E_y$ for $\phi$ and $y$.

6.2. If this was successful, then add the pair $(y, E_y)$ to $M$.

7. If $M = \emptyset$, then

7.1. Insert $(Y, \phi, \emptyset)$ into $F$.

7.2. Return.

8. If variable selection strategy 3, then

8.1. Pick a pair $(y, E_y)$ from $M$ with "the best" elimination set $E_y$.

8.2. For each $(e, \pi, \mathcal{F}) \in E_y$ do

8.2.1. Insert $\big(Y \setminus \{y\}, \phi, y = (e, \pi, \mathcal{F})\big)$ into $W$.

8.3. Return.

9. If variable selection strategy 4, then

9.1. For each pair $(y, E_y) \in M$ do

9.1.1. Substitute the set $E_y$ for $y$ into $\phi$ by means of structural VS.

9.2. Pick the pair $(y, E_y)$ that yielded "the best" resulting formula.

9.3. For each $(e, \pi, \mathcal{F}) \in E_y$ do

9.3.1. Let $\psi$ be the formula obtained by substituting $e$ for $y$ into $\phi_\pi^{\mathcal{F}}$.

9.3.2. Insert $(X \setminus \{y\}, \psi', \emptyset)$ into $W$ for every disjunct $\psi'$ of the formula $\psi$ after simplification.

9.4. Return.

The procedure `expand-node` constructs the children of an internal node in a QE tree. In step 1 we first check whether there exists a variable in $Y$, which does not occur in $\phi$.

Then the procedure `s-shift` of Subsection 4.2.2 tries a structural degree shift w.r.t. each $\pi \in \mathtt{Pos}(\phi)$. Since the resulting nodes are put into $W$ and

eventually considered in some later iteration of `block-loop`, this implicitly simulates the recursive algorithm `s-preproc` discussed below Theorem 77 in Subsection 4.2.2. If $(Y, \phi, s)$ was created from its parent by applying a structural degree shift $y = \left(\sqrt[d]{\tilde{y}}, \pi, \mathcal{F}\right)$, then the recursive depth could be bounded by setting an upper bound on the number of consecutive structural degree shifts w.r.t. shadow variables.

Our implementation does not call `s-shift` in step 2.1 though. It simply tries whether a global degree shift for $y$ and $\phi$ is applicable. Observe that this cannot lead to a consecutive recursive calls as when using `s-preproc`, because a global degree shift is applicable at most once w.r.t. any particular variable. Furthermore, when a global degree shift is applicable, then $\phi^{\mathcal{P}}$ is obviously equivalent to "false" in step 2.2.3.

It is noteworthy that for a "successful" structural degree shift elimination set we obtain in step 2.2.3 a formula $\phi^{\mathcal{P}}$ that does not contain $y$. First, this means that the maximum degree of $\hat{y}$ after shift will be lower than the degree of $y$ in $\phi$. Second, if $\phi^{\mathcal{P}}$ is "false," e.g., when using a global shift, then it can be dropped directly instead of inserting it into $W$. This is what was implicitly done in our example QE tree in Figure 6.1 in node $(\{y, z\}, \varphi_2)$.

In step 3 we know that each variable from $Y$ in fact occurs in $\phi$, and no structural degree shifts are applicable. Therefore, we enter the core of `expand-node`, i.e., a variable selection strategy. We apply one of the following four variable selection strategies:

1. Use the minimum variable $y \in Y$ w.r.t. some prescribed total ordering.
2. Use the first feasible, i.e., "non-failing" variable in $Y$ w.r.t. some prescribed total ordering.
3. Decide which variable from $Y$ to use by looking at the obtained structural elimination sets.
4. Decide which variable from $Y$ to use by eliminating each variable and by looking at the resulting equivalents.

Each computation of a structural elimination set in any of the four strategies is done as we have described in Subsection 6.1.1.

A natural total ordering of the variables used in strategies 1 and 2 satisfies $x_1 \prec \cdots \prec x_n$, whereas all the shadow variables $\hat{x}_{i,j}$ introduced for $x_i$ satisfy $x_i \prec \hat{x}_{i,j} \prec x_{i+1}$. Combined with strategy 1, this means that we eliminate the quantifiers from the inside to the outside in $\exists x_n \ldots \exists x_1(\varphi)$, using as many shadow quantifiers for $x_i$ as needed, eliminating them right after $x_i$.

The default variable selection strategy of the Redlog's original implementation, and also of our implementation, is strategy 3. However, our implementation selects "the best" structural elimination set just by counting the number of test points. Note that this choice can be significantly improved by taking the degrees and the number of other quantified variables and parameters in the computed elimination sets into account.

The greedy strategy 4 might seem to be the best choice among the four strategies. Note that it is the most costly one, and it does not guarantee to obtain the best results after eliminating the whole quantifier block.

An implementation of `block-loop` needs to keep only the actual working nodes and leaves of the tree in memory. Rest of the tree can be deleted by

simply dropping the expanded nodes as we do it tacitly in `block-loop`. If one is interested in obtaining the whole tree, it is straightforward to adjust `block-loop` so that each node contains a reference to its parent. This can be then used to investigate the whole quantifier elimination tree, or as a basis for extended quantifier elimination of Section 5.1 and our EQR post-processing method of Section 5.3.

## 6.3 Computational Experiments

Here we present computational experiments carried out with our implementation whose underlying principles were described in previous sections. The aim is to compare the Redlog's original implementation of quadratic virtual substitution with our implementation both without and with clustering. We are going to compute a number of real quantifier elimination examples coming from three different example sets. To begin with, we discuss each of these example sets.

### Example Sets

The first example set is referred to as "Bath" in the following. This is an example bank of various benchmark problems for CAD-based algorithms.[3] This database has been introduced and its examples thoroughly explained by Wilson in [87]. It contains 78 examples, but only 51 of them are real quantifier elimination problems. The rest are quantifier-free Tarski formulas, and the task in each case is to construct a CAD or a truth table invariant CAD [10] of the real variable space w.r.t. the input formula. For this reason we consider here only the 51 quantifier elimination problems.

The second example set we consider is called "Regressions." These examples are part of the Reduce's source tree,[4] where they are used mainly for daily-build testing and benchmarking. Again, this set contains much more examples testing also other parts of Redlog, but for our purposes here only 19 real quantifier elimination problems among them are relevant.

The third example set "Remis" originates from an example database (Redlog Example Management and Information System) that has been created and is maintained by the senior Redlog developers.[5] The subset of those examples considered here takes all the 154 real quantifier elimination examples from the Remis database.

We point to the fact that each of these three sets consists of relevant real-world examples that have been considered in a wide variety of scientific and engineering fields during the last more than twenty years. Since these examples were thoroughly discussed in more than sixty scientific publications, it is not possible to explain the examples in detail here.

To get an impression about the origin of a few of the examples though, we mention here explicitly five of them, namely: "57-joukowsky-1," "gatermann," "ab07hopf9," "sturm99a_ex3-22," and "linequadbe." These are the names we use to refer to the examples in Appendix B, where we give a thorough listing of results of computational experiments.

[3] V4.0 hosted at `http://www.cs.bath.ac.uk/~djw42/triangular/examplebank.html`

[4] SVN revision 3765 of `svn://svn.code.sf.net/p/reduce-algebra/code/trunk/packages/redlog/regressions/reals`

[5] State as of August 04, 2016, hosted at `http://www.redlog.eu/remis`

The first mentioned example is inspired by a Joukowski conformal map. More precisely, one wants to decide the validity of an identity over the complex plane [87, Chapter 6]. The second example comes from the area of algebraic biology, more precisely from the oscillation analysis of gene regulatory networks [9]. The third example is a highly nonlinear formula coming from the area of Hopf bifurcation analysis considered in [37]. The fourth example comes from the area of automated geometric theorem proving and formalizes a variant of Steiner-Lehmus theorem [67]. Finally, the fifth example originates from the engineering field of computational mechanics, more particularly from the boundary element analysis [43].

For a thorough explanation of each of the examples listed in Appendix B we refer the reader to respective original publications, which can be found as follows:

1. The name of each example in the Bath set consists of an index number and a description shortcut in the CAD example bank, so each example is easily found in the example bank using this information. Consequently, the main reference point for any example from the CAD example bank is the dissertation by Wilson [87]. It discusses the majority of the examples in the example bank itself, and contains references to publications from the CAD community, which originally considered particular examples.

2. The Regressions example set is best explored by downloading the test files from the SVN repository, because the example names correspond with test file names there. These test files contain comments and references to scientific literature they are based on or taken from.

3. Finally, the Remis example set is thoroughly explained on its web page, which contains detailed references and a number of papers ready for download. To obtain detailed information on an example, it suffices to query the web page for the example name.

**Additional Features of the Redlog's Original Implementation**

Before discussing the actual experiments and outcomes, we need to point to substantial differences between our research prototype implementation and the Redlog's mature original implementation of the quadratic virtual substitution.

Since Redlog has been actively developed and improved during the last two decades, it contains a much richer palette of features than our implementation. The features relevant for our experiments here are the following: *positive* real quantifier elimination, *generic* real quantifier elimination, extended real quantifier elimination with and without answers, and combinations thereof.

Positive real quantifier elimination assumes that every variable (a quantified variable as well as a parameter) occurring in the input formula $\varphi$ is strictly positive. This seemingly simple assumption has huge practical impact on the whole quantifier elimination process, and can be exploited intensively to reduce the sizes of elimination sets, speed up virtual substitution procedures, and simplify quantifier-free formulas. Positive real QE is of tremendous importance for examples coming from biology and chemistry, where masses or volumes are often exclusively positive quantities.

Generic real quantifier elimination is allowed to make assumptions of the form $r \neq 0$, where $r \in \mathbb{Z}[\mathbf{u}]$. As a result of a generic QE run on an input formula $\zeta$ one obtains a pair $\left(\{r_i \neq 0\}_i, \psi\right)$ such that $\left(\bigwedge_i r_i \neq 0\right) \longrightarrow (\zeta \longleftrightarrow \psi)$ for a quantifier-free formula $\psi$. In other words $\psi$ is a "generic" quantifier-free equivalent of $\zeta$, i.e., $\psi$ is guaranteed to be equivalent to $\zeta$ only when the parameters have "generic" values satisfying $\bigwedge_i r_i \neq 0$. The method of generic quantifier elimination was originally motivated by automated theorem proving in geometry, where one usually rules out uninteresting situations by statements like "Since three vertices of a triangle do not lie on a line, we w.l.o.g. have..." Something similar is done automatically by generic quantifier elimination by making assumptions that some polynomials from $\mathbb{Z}[\mathbf{u}]$ are nonzero during the quantifier elimination process.

Extended quantifier elimination as a concept has been discussed in Chapter 5, where we have also shown how to post-process nonstandard answers to obtain standard real numbers for fixed values of the parameters. Moreover, we have reported on our successful implementation of the post-processing method within the quadratic quantifier elimination in Redlog's original implementation.

### Computations and Results

All of the Redlog additional features discussed above have been successfully applied to many examples in our example sets. However, to make the comparison with our implementation—that does not support any of these features—meaningful, we simply modified the examples to use only the standard quadratic quantifier elimination, i.e., to always call only `rlqe` instead of other calls of the additional Redlog features.

Furthermore, some examples in the example sets Regressions and Remis were originally records of interactive Redlog computations containing a number of successive quantifier elimination calls. In such cases we manually extracted the formulas that enter quantifier elimination. This often meant that some simplifications were not done, or that more quantifiers were left in an input formula for elimination. Of course, in each case a formula in our example set is equivalent to its corresponding formula in the original example. In this way we have obtained by our manual extraction, e.g., from "sw97_ex12" two examples, namely "sw97_ex12-1" and "sw97_ex12-2," as we can see in Appendix A.

Observe that with the Bath example set we simply had to convert the formula from the Qepcad input format into the Redlog input format, because the Qepcad input format allows one to provide exactly one prenex formula on the input.

After all this preprocessing we finally obtained our homogeneous sets of examples, where each example is a Redlog file containing *exactly* one input Tarski formula in the prenex normal form. The task of all three examined implementations is to eliminate the quantifiers without using any additional features, i.e., by using only the standard real quantifier elimination by virtual substitution.

All computations have been carried out with the CSL variant of Reduce, revision 3765, using 4 GB RAM on a 2.4 GHz Intel Xeon E5-4640 running 64 bit Debian Linux 8.3 with the time limit of one hour user time per example. Since our implementation is a part of Redlog, it is sufficient to go `on ofsfvs;` in a Redlog session to activate its use instead of the Redlog's original implementation, which is in use by default. Note that clustering is on by default in our implementation;

| | Original | Without Clustering | With Clustering |
|---|---|---|---|
| | | Bath Example Set | |
| Timeout | 0 | 0 | 0 |
| Fail | 6 | 2 | 2 |
| Partial | 8 | 11 | 11 |
| Success | 37 | 38 | 38 |
| | | Regressions Example Set | |
| Timeout | 0 | 0 | 0 |
| Fail | 2 | 2 | 2 |
| Partial | 3 | 3 | 2 |
| Success | 14 | 14 | 15 |
| | | Remis Example Set | |
| Timeout | 10 | 7 | 6 |
| Fail | 9 | 5 | 5 |
| Partial | 7 | 8 | 6 |
| Success | 128 | 134 | 137 |

Table 6.1: Summary of results of the computational experiments carried out on the example sets Bath, Regressions, and Remis containing 51, 19, and 154 examples, respectively.

to turn it off, one executes `symbolic; rlclustering!* := nil; algebraic;`. To measure in each case only the performance of the quantifier elimination by virtual substitution, we furthermore disabled the Redlog's default feature to call a "fallback" QE method—which is usually a CAD algorithm—when virtual substitution fails. This is done by executing `off rlqefb;` in a Redlog session.

Table 6.1 gives a summary of the computational results; detailed information can be found in Appendix A. The columns "Original," "Without Clustering," and "With Clustering" refer to the default Redlog's original implementation of quadratic quantifier elimination, our implementation without and with clustering, respectively.

For each example set and implementation, we divided the examples into four groups: The "Timeout" row shows the number of examples on which an implementation did not terminate within the one hour user time limit. The "Fail" row lists the number of examples on which an implementation returned a formula with the same number of quantifiers as the input formula, i.e., the degrees of all the quantified variables in the input formula were too high for an implementation to eliminate. The "Partial" row shows the number of examples on which at least one quantifier was successfully eliminated. Finally, the "Success" row contains the number of examples where an implementation was able to eliminate all quantifiers from the input formula.

The main message of Table 6.1 is that our implementation with clustering never eliminates fewer quantifiers than the Redlog's original implementation. Moreover, a careful look at the full listing in Appendix A reveals that for *every* example our implementation with clustering was able to eliminate at least as many quantifiers as the Redlog's original implementation. Moreover, in

cases when some quantifiers were left, our implementation left the same or even *smaller* number of quantifiers than the Redlog's original implementation.

At the same time, we point to the fact that the numbers of atomic formulas of the resulting formulas with the same number of remaining quantifiers differ dramatically in some cases. A few extreme cases are:

- sw97_ex20-2, where the Redlog's original implementation returned a resulting quantifier-free formula with 12,313 atomic formulas, whereas our implementation with clustering returned a quantifier-free equivalent with 1,792 atoms,
- netan1-09, where the Redlog's original implementation returned a resulting quantifier-free equivalent with 650 atoms, and the result obtained by our implementation with clustering contained only 5 atoms,
- ab07hopf9, where the Redlog's original implementation returned a formula with two quantifiers left having 2,463 atoms, whereas our implementation with clustering yielded a formula with two quantifiers and 30,143 atomic formulas,
- dsw96_ex10, where with clustering we obtained a quantifier-free equivalent with 815 atoms, whereas without clustering we computed a quantifier-free formula with 20,497 atomic formulas,
- sw97_ex17, where our implementation with clustering yielded a quantifier-free equivalent with 22,608 atoms, whereas without clustering it computed a quantifier-free equivalent with only 572 atoms.

The reason that the resulting formulas obtained by our implementation are sometimes longer is most probably the fact that our implementation uses a simple variable selection heuristic and explores a completely different QE tree than the Redlog's original implementation.

The most probable reason why our implementation without clustering sometimes returns shorter formulas than with clustering is that longer non-clustered formulas possibly allow for more simplifications, which can be sometimes beneficial and greatly exploited by the simplifier.

Comparing the overall running time of our implementation using clustering with the running time of the Redlog's original implementation on examples where both implementations terminated within the one hour user time limit we observed that our implementation is slower by factor $\sim$3.4. Most probably this is caused by numerous factorizations, multiple formula traversals, and more complex virtual substitution procedures. Besides these, one also has to take into account that the implementations construct different QE trees containing different subproblems allowing different simplifications. To precisely compare the running time of the implementations, one would need to compare the running time on *the same* QE trees, which is in principle not possible.

To sum up, our experiments point to a great practical potential of the methods presented in this thesis. Observe that our implementation essentially implements only virtual substitution for degree three, clustering, and structural virtual substitution. More sophisticated degree shift techniques, bound selection strategies, or variable selection heuristics were not implemented at all. Despite these obvious limitations, our implementation never eliminates fewer quantifiers,

and often outperforms the Redlog's original implementation—which has been developed and improved over the last two decades—in the number of eliminated quantifiers on the problem sets considered in this section. This strongly suggest that implementations extending our implementation with the mentioned enhancements and optimizations along with features like positive or generic quantifier elimination will significantly shift the boundary of applicable real quantifier elimination in the sciences and engineering. Of course, this is left as the most exciting future direction of this thesis.

## 6.4 Conclusions

In this chapter we have discussed practical issues one faces when implementing a virtual substitution-based quantifier elimination algorithm. We have shown how to efficiently implement some of the theoretical concepts presented earlier in this thesis. Based on our structural virtual substitution scheme of Chapter 3, we have discussed how to implement the elimination of one existential quantifier and consequently of one existential quantifier block. We have shown on more than 200 relevant real-world quantifier elimination examples that our implementation relying on the principles discussed in this chapter outperforms the Redlog's original implementation of the quadratic virtual substitution in the following sense:

1. Our implementation *never* eliminates fewer quantifiers, and
2. in a number of cases it eliminates even more quantifiers.

This shows a great practical potential of the theoretical concepts developed in this thesis.

# Chapter 7

# Conclusions and Future Directions

## 7.1 Summary

We have studied methods for effective quantifier elimination in real closed fields based on virtual substitution.

We have devised a uniform framework for real quantifier elimination based on virtual substitution, which can be instantiated to arbitrary but bounded degrees of quantified variables. Instantiation for a particular degree requires only three algorithms for computing candidate solutions, guards, and performing virtual substitution into an atomic formula, respectively. We have provided such instantiations until degree three. Our work comprises numerous improvements, like smaller elimination sets and clustering, which are applicable to and significantly improve even the traditional approaches with root expressions due to Weispfenning [84].

In particular, we have shown how to take advantage of the Boolean structure of an input formula using a structural quantifier elimination algorithm scheme based on the concepts of Gauss and co-Gauss subformulas, prime constituents, and prime constituent decompositions. We have shown how the Boolean structure can be used during substitution of a test point and during bound selection to obtain smaller elimination sets compared to approaches ignoring the Boolean structure. Our framework allowed us to reformulate as virtual substitution and to even generalize heuristic of degree shifts, which decreases the degrees of quantified variables in certain cases. We have made explicit limitations of that heuristic technique by showing that a systematic exploitation of the idea can in rare cases lead to naive DNF computation of the input formula.

For existential problems, we have described extended quantifier elimination within our framework. The approach of extended quantifier elimination yields in addition to quantifier-free equivalents also answers for the quantified variables. On these grounds we have generalized a recent post-processing technique for eliminating nonstandard symbols from answers, which we had presented for the quadratic case in [47].

An implementation of our framework provides an extension of the computer logic system Redlog with quantifier elimination by virtual substitution that is

complete up to degree three of a quantified variable. We have discussed here practical implementation aspects including data structures for elimination of a block of quantifiers, prime constituent decomposition, and lazy structural virtual substitution. We have conducted and interpreted comprehensive computational experiments, which demonstrate the relevance of our theoretical concepts and the efficiency of our implementation. The results are encouraging: Our research prototype implementation never eliminates fewer quantifiers than the Redlog's original implementation of quadratic virtual substitution. In many cases of the real quantifier elimination example instances, we were able to eliminate more quantifiers than the Redlog's original implementation.

## 7.2 Some Future Directions

The difficulties of the tasks discussed below range from technical implementation improvements to open research questions. We organize the material according to related parts of the thesis.

#### Virtual Substitution Framework

In Subsection 2.5.4 we have discussed how to derive guards and virtual substitution formula schemes for degree four. The next step is to actually derive full virtual substitution formula schemes as with the quadratic and cubic cases in Appendix A. Recall that thanks to the modularity of our approach, integration into our implementation would then be straightforward.

The central idea with the derivation of virtual substitution formula schemes in Subsection 2.5.4 is to consider the mutual geometric position of two polynomials to derive necessary and sufficient conditions on $g \varrho 0$ to hold at a particular root of $f$. As discussed there, the admissible regions of some particular root of $f$ for $g \varrho 0$ to hold is described by sign conditions on the derivatives of $f$ and $g$. It would be a great step forward to produce practically useful sign conditions automatically. This is in principle possible thanks to real quantifier elimination but infeasible in practice. An efficient special-purpose algorithm would then represent a machine support for the production of implementations for increasing degrees, which could be combined with subsequent human optimizations.

It is by no means clear whether the formula schemes of Appendix A are of optimal length. Replacing them with shorter quantifier-free equivalents, if possible, by combining automated methods and human intelligence is a natural step that would significantly improve the performance of all virtual substitution algorithms presented in this thesis. It is noteworthy that in the past, mathematicians have been attracted to finding optimal solutions to particular quantifier elimination problems, e.g., Kahan's Problem [50]. Focusing those techniques and efforts on our quantifier elimination formula schemes would improve not only a single result but a powerful procedure for automatically producing such results.

The principal difference between our test points, which are essentially parametric root descriptions, and the elimination terms in the context of quadratic and linear virtual substitution is that the latter are viewed as terms in some extension language $\mathfrak{L}'$ of the Tarski language $\mathfrak{L}$. It is clear that test points within our framework could be formalized as terms in a suitable extension lan-

guage of $\mathfrak{L}$ as well. However, for practical applications it is important to find a "natural" extension language that allows arithmetic or normal form computations of our test points regarded then as terms. This would allow us to carry out back-substitution similarly to our discussion in Section 5.2. Furthermore, computing from two parametric root descriptions a parametric root description representing their arithmetic mean could be similar to what Weispfenning [81] did to avoid infinitesimals in an early version of the linear virtual substitution, or what Dolzmann [25, Section 3.7] did when considering formulas with only one nonlinear quadratic constraint. Suitable extended terms would allow to lift these approaches to higher degrees and try them in practice.

It would be interesting to try and adjust our algorithms to operate on nonstandard expressions directly, i.e., without calling an expansion algorithm like `expand-eps-at` before substituting $(f, S) \pm \varepsilon$ or $\pm\infty$. Instead one would simply substitute $(f, S) \pm \varepsilon$ and $\pm\infty$ by means of some virtual substitution formula schemes, obtaining formulas possibly containing nonstandard symbols. An approach working with nonstandard symbols would allow to eliminate all quantifiers to obtain a quantifier-free equivalent over a nonstandard extension field of $\mathbb{R}$. Such nonstandard quantifier-free equivalent would afterwards have to be equivalently rewritten into a Tarski formula. It should be investigated whether this is practically useful, what simplifications of quantifier-free formulas would be feasible in the presence of nonstandard symbols, and how one could efficiently rewrite such nonstandard formulas into Tarski equivalents.

### Structural Virtual Substitution

In [25, Section 3.7] Dolzmann considered formulas with only one nonlinear quadratic constraint. He proposed elimination sets using the derivative of that quadratic constraint along with other linear constraints, preventing the use of quadratic elimination terms altogether.

First, we believe that this can be easily extended to the cubic and higher degree cases. Second, it would be interesting to what extent this could be combined with our concept of structural virtual substitution and conjunctive associativity. This could yield concepts like special cases w.r.t. a subformula or special cases w.r.t. other conjunctively associated formulas in a formula, possibly generating elimination sets containing test points of lower degrees.

### Degree Shift

In Section 4.2 we conjectured that the degree shift approach trying all positions also constructs the DNF of certain "bad" formulas $\phi$. However, Lemma 75 used in the proof of this fact for `s-prepoc-at` does *not* hold. Possible future work in this direction would be to prove that $\phi$ is "bad" also for `s-preproc`, or to come up with some other "bad" formulas for which `s-preproc` computes the DNF.

In Section 4.3 we constructed "bad" formulas $\phi$ that allow for structural degree shifts lowering the degree of a quantified variable only after full DNF computation. The degrees of $x$ in these formulas were extremely high, so it would be interesting to asymptotically estimate the bit complexity of these formulas using the Prime Number Theorem. Furthermore, it would be interesting to investigate whether there exist shorter "bad" formulas.

### Answers for Virtual Substitution

Recall from our discussion in Section 5.4 on finding simple parametric root descriptions representing rationals or even integers that there is often a considerable degree of freedom in how we achieve this. In the future this can be further exploited in various interesting ways: For instance, using extended quantifier elimination methods as a theory solver in the context of Satisfiability Modulo Theory (SMT) solving [58], in particular when combining several theories in a Nelson–Oppen [56] style, one is specifically interested in avoiding identical answers for different variables.

A theoretically way more challenging step would be the generalization of the method of Chapter 5 to the parametric case. Recall that Proposition 86 has shown that it is not possible in general to determine constant real standard values for nonstandard symbols to straightforwardly transform the nonstandard answers to standard answers. Nevertheless, it might well be possible to devise on the basis of our work a complete method for symbolically replacing nonstandard answers with standard ones. In the example we considered in Proposition 86 the answer $x = \big(x - 1, (1, 1)\big) - \varepsilon_1$ could be replaced, e.g., with a standard answer $x = \big(2x - u - 1, (1, 1)\big)$ yielding a standard extended quantifier elimination result.

Another theoretical challenge would be to generalize the concept of extended quantifier elimination and our answer correcting procedure for richer quantifier prefixes by allowing at least one quantifier alternations. For existential variables we should be then able to obtain from our quantifier elimination procedure answers as "Skolem functions" of parameters and universally quantified variables.

### Implementation

There is a big number of possible future developments of our implementation described in Chapter 6. We therefore mention here only a few of them.

The easiest improvement is probably to hook into the traversal of the formula when substituting an elimination term $e$ into $\varphi$. One could save some time consuming virtual substitutions by traversing the Boolean structure of $\varphi$ and stop substituting when after simplification "true" or "false" are with "$\vee$" or "$\wedge$," respectively.

Our implementation does not yet support the 0-1 ILP-based bound selection strategy discussed in Section 3.4, so a possible next step is to implement it and integrate it into our implementation.

Our variable selection heuristic is currently very simple: We merely count the number of test points in an elimination set. Here one could definitely do better by preferring linear to quadratic to cubic test points. Such a detection is already a part of the Redlog's original implementation. Moreover, analyzing other variables occurring in the parametric root descriptions during the comparison of elimination sets, one should prefer elimination sets containing fewer quantified variables and with variables of lower degrees.

In [25, Chapter 4], Dolzmann introduced various theory concepts in the context of simplification and quantifier elimination. A *theory* is here a set of Tarski atomic formulas, which is assumed to holds. The various concepts can tremendously simplify computed elimination sets or resulting quantifier-free equivalents obtained by virtual substitution. Parts of these concepts have been successfully implemented in the Redlog's original implementation.

The features of the Redlog's original implementation we had to disable for our computational experiments in Section 6.3 like positive or generic quantifier elimination should definitely be added to our implementation. These techniques are known to be crucial for the practical applicability of quantifier elimination by virtual substitution with many problems from the sciences and engineering.

One effect of our smaller elimination sets and bound selection strategies is that we can often strengthen a formula $a \neq 0$ occurring in a guard to $a \gtrless 0$, i.e., we know the sign of $a$. In contrast, the Redlog's original implementation always takes $a \neq 0$. Using this fact could tremendously simplify subformulas, e.g., by factoring out $a$ where possible. However, one would have to adjust the simplifier of quantifier-free Tarski formulas. Redlog's simplifier is namely tuned to work optimally in cooperation with the Redlog's original implementation.

To conclude we would like to emphasize once more that in spite lacking all features discussed above our implementation already competes well with the Redlog's original mature implementation of virtual substitution. Therefore, we are confident that a more elaborate implementation of the novel framework developed in this thesis will significantly improve the applicability of real quantifier elimination in computer science, engineering, and the sciences.

# Appendix A

# Formula Schemes for Virtual Substitution

In this appendix we list all virtual substitution formula schemes necessary to realize algorithms `vs-prd-at-2` and `vs-prd-at-3` given in Chapter 2. Note that here we also use the convention introduced in Section 2.5: Instead of referring to a real $d$-type $t$ by its full sequence of signs, we abbreviate this and use the respective number of $t$ in the list of real types given at the beginning of Subsection 2.5.2.

## A.1 The Quadratic Case

Let $f = ax^2 + bx + c$, where $a$, $b$, $c \in \mathbb{Z}[\mathbf{u}]$.

#### Target Atomic Formula of Degree One

Let $g = a^*x + b^*$, where $a^*$, $b^* \in \mathbb{Z}[\mathbf{u}]$. Then we have:

$(g = 0)\big[x /\!\!/ (f, (1,1))\big]$:

$$2aa^*b^* - {a^*}^2 b \geq 0 \wedge ab^{*2} + {a^*}^2 c - a^* b b^* = 0$$

$(g = 0)\big[x /\!\!/ (f, (1,2))\big]$:

$$2aa^*b^* - {a^*}^2 b \leq 0 \wedge ab^{*2} + {a^*}^2 c - a^* b b^* = 0$$

$(g = 0)\big[x /\!\!/ (f, (2,1))\big]$:

$$2ab^* - a^*b = 0$$

---

$(g < 0)\big[x /\!\!/ (f, (1,1))\big]$:

$$\begin{aligned} &2ab^* - a^*b < 0 \wedge ab^{*2} + {a^*}^2 c - a^* b b^* > 0 \vee \\ &a^* \geq 0 \wedge (2ab^* - a^*b < 0 \vee ab^{*2} + {a^*}^2 c - a^* b b^* < 0) \end{aligned}$$

$(g < 0)\big[x \mathbin{/\!\!/} (f,(1,2))\big]$:

$$2ab^* - a^*b < 0 \wedge {ab^*}^2 + {a^*}^2 c - a^* bb^* > 0 \vee$$
$$a^* \leq 0 \wedge (2ab^* - a^*b < 0 \vee {ab^*}^2 + {a^*}^2 c - a^* bb^* < 0)$$

$(g < 0)\big[x \mathbin{/\!\!/} (f,(2,1))\big]$:

$$2ab^* - a^*b < 0$$

---

$(g \leq 0)\big[x \mathbin{/\!\!/} (f,(1,1))\big]$:

$$2ab^* - a^*b \leq 0 \wedge {ab^*}^2 + {a^*}^2 c - a^* bb^* \geq 0 \vee$$
$$a^* \geq 0 \wedge {ab^*}^2 + {a^*}^2 c - a^* bb^* \leq 0$$

$(g \leq 0)\big[x \mathbin{/\!\!/} (f,(1,2))\big]$:

$$2ab^* - a^*b \leq 0 \wedge {ab^*}^2 + {a^*}^2 c - a^* bb^* \geq 0 \vee$$
$$a^* \leq 0 \wedge {ab^*}^2 + {a^*}^2 c - a^* bb^* \leq 0$$

$(g \leq 0)\big[x \mathbin{/\!\!/} (f,(2,1))\big]$:

$$2ab^* - a^*b \leq 0$$

## A.2 The Cubic Case

Let $f = ax^3 + bx^2 + cx + d$, where $a$, $b$, $c$, $d \in \mathbb{Z}[\mathbf{u}]$. In the following we denote:

$$\begin{aligned} f' &= 3ax^2 + 2bx + c, \\ f'' &= 6ax + 2b, \\ \alpha_1 &= (f', (1,1)), \\ \alpha_2 &= (f', (1,2)). \end{aligned}$$

**Target Atomic Formula of Degree One**

Let $g = a^*x + b^*$ be such that $a^* > 0$. In the following we denote $\beta = (g,(1,1))$. Then we have:

$(g = 0)\big[x \mathbin{/\!\!/} (f,(1,1))\big]$:

$$(f = 0)[x \mathbin{/\!\!/} \beta]$$

$(g = 0)\big[x \mathbin{/\!\!/} (f,(2,1))\big]$:

$$(g = 0)[x \mathbin{/\!\!/} \alpha_1]$$

$(g = 0)\big[x \mathbin{/\!\!/} (f,(2,2))\big]$:

$$(f = 0)[x \mathbin{/\!\!/} \beta] \wedge (g \leq 0)[x \mathbin{/\!\!/} \alpha_2]$$

$(g = 0)\big[x \mathbin{/\!\!/} (f,(3,1))\big]$:

$$(f = 0)[x \mathbin{/\!\!/} \beta] \wedge (g \geq 0)[x \mathbin{/\!\!/} \alpha_1]$$

$(g = 0)\big[x \mathbin{/\!\!/} (f,(3,2))\big]$:

$$(g = 0)[x \mathbin{/\!\!/} \alpha_2]$$

$(g = 0)\big[x /\!\!/ (f,(4,1))\big]$:

$$(f = 0)[x /\!\!/ \beta] \wedge (g \geq 0)[x /\!\!/ \alpha_1]$$

$(g = 0)\big[x /\!\!/ (f,(4,2))\big]$:

$$(f = 0)[x /\!\!/ \beta] \wedge (g \leq 0)[x /\!\!/ \alpha_1] \wedge (g \geq 0)[x /\!\!/ \alpha_2]$$

$(g = 0)\big[x /\!\!/ (f,(4,3))\big]$:

$$(f = 0)[x /\!\!/ \beta] \wedge (g \leq 0)[x /\!\!/ \alpha_2]$$

---

$(g < 0)\big[x /\!\!/ (f,(1,1))\big]$:

$$(f > 0)[x /\!\!/ \beta]$$

$(g < 0)\big[x /\!\!/ (f,(2,1))\big]$:

$$(g < 0)[x /\!\!/ \alpha_1]$$

$(g < 0)\big[x /\!\!/ (f,(2,2))\big]$:

$$(f > 0)[x /\!\!/ \beta]$$

$(g < 0)\big[x /\!\!/ (f,(3,1))\big]$:

$$(f > 0)[x /\!\!/ \beta] \vee (g = 0)[x /\!\!/ \alpha_2]$$

$(g < 0)\big[x /\!\!/ (f,(3,2))\big]$:

$$(g < 0)[x /\!\!/ \alpha_2]$$

$(g < 0)\big[x /\!\!/ (f,(4,1))\big]$:

$$(f > 0)[x /\!\!/ \beta] \vee (g \leq 0)[x /\!\!/ \alpha_1]$$

$(g < 0)\big[x /\!\!/ (f,(4,2))\big]$:

$$(g \leq 0)[x /\!\!/ \alpha_2] \vee (f < 0)[x /\!\!/ \beta] \wedge (g \leq 0)[x /\!\!/ \alpha_1]$$

$(g < 0)\big[x /\!\!/ (f,(4,3))\big]$:

$$(f > 0)[x /\!\!/ \beta] \wedge (g \leq 0)[x /\!\!/ \alpha_2]$$

---

$(g \leq 0)\big[x /\!\!/ (f,(1,1))\big]$:

$$(f \geq 0)[x /\!\!/ \beta]$$

$(g \leq 0)\big[x /\!\!/ (f,(2,1))\big]$:

$$(g \leq 0)[x /\!\!/ \alpha_1]$$

$(g \leq 0)\big[x /\!\!/ (f,(2,2))\big]$:

$$(f \geq 0)[x /\!\!/ \beta] \wedge (g \leq 0)[x /\!\!/ \alpha_2]$$

$(g \leq 0)\big[x /\!\!/ (f,(3,1))\big]$:

$$(f \geq 0)[x /\!\!/ \beta]$$

$(g \leq 0)\big[x /\!\!/ (f,(3,2))\big]$:

$$(g \leq 0)[x /\!\!/ \alpha_2]$$

$(g \le 0)\big[x /\!\!/ (f,(4,1))\big]$:

$$(f \ge 0)[x /\!\!/ \beta] \vee (g \le 0)[x /\!\!/ \alpha_1]$$

$(g \le 0)\big[x /\!\!/ (f,(4,2))\big]$:

$$(g \le 0)[x /\!\!/ \alpha_2] \vee (f \le 0)[x /\!\!/ \beta] \wedge (g \le 0)[x /\!\!/ \alpha_1]$$

$(g \le 0)\big[x /\!\!/ (f,(4,3))\big]$:

$$(f \ge 0)[x /\!\!/ \beta] \wedge (g \le 0)[x /\!\!/ \alpha_2]$$

**Target Atomic Formula of Degree Two**

Let $g = a^*x^2 + b^*x + c^*$, where $a^*$, $b^*$, $c^* \in \mathbb{Z}[\mathbf{u}]$. Assume that $a^* > 0$, and denote $D_g = {b^*}^2 - 4a^*c^*$, $\beta_1 = (g,(1,1))$, $\beta_2 = (g,(1,2))$. Then we have:

$(g = 0)\big[x /\!\!/ (f,(1,1))\big]$:

$$D_g \ge 0 \wedge \big((f = 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2]\big)$$

$(g = 0)\big[x /\!\!/ (f,(2,1))\big]$:

$$(g = 0)[x /\!\!/ \alpha_1]$$

$(g = 0)\big[x /\!\!/ (f,(2,2))\big]$:

$$D_g \ge 0 \wedge \big((f = 0)[x /\!\!/ \beta_1] \wedge (f'' \ge 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2] \wedge (f'' \ge 0)[x /\!\!/ \beta_2]\big)$$

$(g = 0)\big[x /\!\!/ (f,(3,1))\big]$:

$$D_g \ge 0 \wedge \big((f = 0)[x /\!\!/ \beta_1] \wedge (f'' \le 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2] \wedge (f'' \le 0)[x /\!\!/ \beta_2]\big)$$

$(g = 0)\big[x /\!\!/ (f,(3,2))\big]$:

$$(g = 0)[x /\!\!/ \alpha_2]$$

$(g = 0)\big[x /\!\!/ (f,(4,1))\big]$:

$$D_g \ge 0 \wedge \big((f = 0)[x /\!\!/ \beta_1] \wedge (f' \ge 0)[x /\!\!/ \beta_1] \wedge (f'' \le 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2] \wedge (f' \ge 0)[x /\!\!/ \beta_2] \wedge (f'' \le 0)[x /\!\!/ \beta_2]\big)$$

$(g = 0)\big[x /\!\!/ (f,(4,2))\big]$:

$$D_g \ge 0 \wedge \big((f = 0)[x /\!\!/ \beta_1] \wedge (f' \le 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2] \wedge (f' \le 0)[x /\!\!/ \beta_2]\big)$$

$(g = 0)\big[x /\!\!/ (f,(4,3))\big]$:

$$D_g \ge 0 \wedge \big((f = 0)[x /\!\!/ \beta_1] \wedge (f' \ge 0)[x /\!\!/ \beta_1] \wedge (f'' \ge 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2] \wedge (f' \ge 0)[x /\!\!/ \beta_2] \wedge (f'' \ge 0)[x /\!\!/ \beta_2]\big)$$

$(g<0)\big[x /\!\!/ (f,(1,1))\big]$:

$$D_g>0\wedge(f<0)[x /\!\!/ \beta_1]\wedge(f>0)[x /\!\!/ \beta_2]$$

$(g<0)\big[x /\!\!/ (f,(2,1))\big]$:

$$(g<0)[x /\!\!/ \alpha_1]$$

$(g<0)\big[x /\!\!/ (f,(2,2))\big]$:

$$D_g>0\wedge\big((f<0)[x /\!\!/ \beta_1]\vee(f'=0)[x /\!\!/ \beta_1]\big)\wedge(f>0)[x /\!\!/ \beta_2]$$

$(g<0)\big[x /\!\!/ (f,(3,1))\big]$:

$$D_g>0\wedge(f<0)[x /\!\!/ \beta_1]\wedge\big((f>0)[x /\!\!/ \beta_2]\vee(f'=0)[x /\!\!/ \beta_2]\big)$$

$(g<0)\big[x /\!\!/ (f,(3,2))\big]$:

$$(g<0)[x /\!\!/ \alpha_2]$$

$(g<0)\big[x /\!\!/ (f,(4,1))\big]$:

$$D_g>0\wedge\big((f<0)[x /\!\!/ \beta_1]\wedge(f'\geq 0)[x /\!\!/ \beta_1]\wedge(f''\leq 0)[x /\!\!/ \beta_1]\big)\wedge \big((f>0)[x /\!\!/ \beta_2]\vee(g\leq 0)[x /\!\!/ \alpha_1]\big)$$

$(g<0)\big[x /\!\!/ (f,(4,2))\big]$:

$$D_g>0\wedge\big((f>0)[x /\!\!/ \beta_1]\vee(g\leq 0)[x /\!\!/ \alpha_1]\big)\wedge \big((f<0)[x /\!\!/ \beta_2]\vee(g\leq 0)[x /\!\!/ \alpha_2]\big)$$

$(g<0)\big[x /\!\!/ (f,(4,3))\big]$:

$$D_g>0\wedge\big((f<0)[x /\!\!/ \beta_1]\vee(g\leq 0)[x /\!\!/ \alpha_2]\big)\wedge \big((f>0)[x /\!\!/ \beta_2]\wedge(f'\geq 0)[x /\!\!/ \beta_2]\wedge(f''\geq 0)[x /\!\!/ \beta_2]\big)$$

---

$(g\leq 0)\big[x /\!\!/ (f,(1,1))\big]$:

$$D_g\geq 0\wedge(f\leq 0)[x /\!\!/ \beta_1]\wedge(f\geq 0)[x /\!\!/ \beta_2]$$

$(g\leq 0)\big[x /\!\!/ (f,(2,1))\big]$:

$$(g\leq 0)[x /\!\!/ \alpha_1]$$

$(g\leq 0)\big[x /\!\!/ (f,(2,2))\big]$:

$$D_g\geq 0\wedge(f\leq 0)[x /\!\!/ \beta_1]\wedge(f\geq 0)[x /\!\!/ \beta_2]\wedge(f''\geq 0)[x /\!\!/ \beta_2]$$

$(g\leq 0)\big[x /\!\!/ (f,(3,1))\big]$:

$$D_g\geq 0\wedge(f\leq 0)[x /\!\!/ \beta_1]\wedge(f''\leq 0)[x /\!\!/ \beta_1]\wedge(f\geq 0)[x /\!\!/ \beta_2]$$

$(g\leq 0)\big[x /\!\!/ (f,(3,2))\big]$:

$$(g\leq 0)[x /\!\!/ \alpha_2]$$

$(g \le 0)\big[x /\!\!/ (f,(4,1))\big]$:

$$D_g \ge 0 \wedge \big((f \le 0)[x /\!\!/ \beta_1] \wedge (f' \ge 0)[x /\!\!/ \beta_1] \wedge (f'' \le 0)[x /\!\!/ \beta_1]\big) \wedge \big((f \ge 0)[x /\!\!/ \beta_2] \vee (g \le 0)[x /\!\!/ \alpha_1]\big)$$

$(g \le 0)\big[x /\!\!/ (f,(4,2))\big]$:

$$D_g \ge 0 \wedge \big((f \ge 0)[x /\!\!/ \beta_1] \wedge (f' \le 0)[x /\!\!/ \beta_1] \vee (g \le 0)[x /\!\!/ \alpha_1]\big) \wedge \big((f \le 0)[x /\!\!/ \beta_2] \vee (g \le 0)[x /\!\!/ \alpha_2]\big)$$

$(g \le 0)\big[x /\!\!/ (f,(4,3))\big]$:

$$D_g \ge 0 \wedge \big((f \le 0)[x /\!\!/ \beta_1] \vee (g \le 0)[x /\!\!/ \alpha_2]\big) \wedge \big((f \ge 0)[x /\!\!/ \beta_2] \wedge (f' \ge 0)[x /\!\!/ \beta_2] \wedge (f'' \ge 0)[x /\!\!/ \beta_2]\big)$$

## A.3 The Quadratic Case with Clustering

Let $f = ax^2 + bx + c$, where $a$, $b$, $c \in \mathbb{Z}[\mathbf{u}]$.

**Target Atomic Formula of Degree One**

Let $g = a^*x + b^*$, where $a^*$, $b^* \in \mathbb{Z}[\mathbf{u}]$. Then we have:

$(a^*x + b^* = 0)\big[x /\!\!/ (f,\{(1,1),(2,1),(-1,2),(-2,1)\})\big]$:

$$2aa^*b^* - a^{*2}b \ge 0 \wedge ab^{*2} + a^{*2}c - a^*bb^* = 0$$

$(a^*x + b^* = 0)\big[x /\!\!/ (f,\{(1,2),(2,1),(-1,1),(-2,1)\})\big]$:

$$2aa^*b^* - a^{*2}b \le 0 \wedge ab^{*2} + a^{*2}c - a^*bb^* = 0$$

---

$(a^*x + b^* < 0)\big[x /\!\!/ (f,\{(1,1),(2,1),(-1,2),(-2,1)\})\big]$:

$$2a^2b^* - aa^*b < 0 \wedge a^2b^{*2} + aa^{*2}c - aa^*bb^* > 0 \vee aa^* \ge 0 \wedge (2a^2b^* - aa^*b < 0 \vee a^2b^{*2} + aa^{*2}c - aa^*bb^* < 0)$$

$(a^*x + b^* < 0)\big[x /\!\!/ (f,\{(1,2),(2,1),(-1,1),(-2,1)\})\big]$:

$$2a^2b^* - aa^*b < 0 \wedge a^2b^{*2} + aa^{*2}c - aa^*bb^* > 0 \vee aa^* \le 0 \wedge (2a^2b^* - aa^*b < 0 \vee a^2b^{*2} + aa^{*2}c - aa^*bb^* < 0)$$

---

$(a^*x + b^* \le 0)\big[x /\!\!/ (f,\{(1,1),(2,1),(-1,2),(-2,1)\})\big]$:

$$2a^2b^* - aa^*b \le 0 \wedge a^2b^{*2} + aa^{*2}c - aa^*bb^* \ge 0 \vee aa^* \ge 0 \wedge a^2b^{*2} + aa^{*2}c - aa^*bb^* \le 0$$

$(a^*x + b^* \le 0)\big[x /\!\!/ (f,\{(1,2),(2,1),(-1,1),(-2,1)\})\big]$:

$$2a^2b^* - aa^*b \le 0 \wedge a^2b^{*2} + aa^{*2}c - aa^*bb^* \ge 0 \vee aa^* \le 0 \wedge a^2b^{*2} + aa^{*2}c - aa^*bb^* \le 0$$

## A.4 The Cubic Case with Clustering

Let $f = ax^3 + bx^2 + cx + d$, where $a$, $b$, $c$, $d \in \mathbb{Z}[\mathbf{u}]$. In the following we denote:

$$\begin{aligned} f' &= 3ax^2 + 2bx + c, \\ f'' &= 6ax + 2b, \\ \alpha_1 &= (f', (1,1)), \\ \alpha_2 &= (f', (1,2)). \end{aligned}$$

**Target Atomic Formula of Degree One**

Let $g = a^*x + b^*$ be such that $a^*$, $b^* \in \mathbb{Z}[\mathbf{u}]$ and $a^* > 0$. Let $\beta = (g, (1,1))$. Then we have:

$(g = 0)\big[x \mathbin{/\!\!/} (f, (1,1))\big]$:

$$(f = 0)[x \mathbin{/\!\!/} \beta]$$

$(g = 0)\big[x \mathbin{/\!\!/} (f, \{(2,1), (3,1), (4,1)\})\big]$:

$$(f = 0)[x \mathbin{/\!\!/} \beta] \wedge (g \geq 0)[x \mathbin{/\!\!/} \alpha_1]$$

$(g = 0)\big[x \mathbin{/\!\!/} (f, \{(2,1), (3,2), (4,2)\})\big]$:

$$(f = 0)[x \mathbin{/\!\!/} \beta] \wedge (f' \leq 0)[x \mathbin{/\!\!/} \beta]$$

$(g = 0)\big[x \mathbin{/\!\!/} (f, \{(2,2), (3,2), (4,3)\})\big]$:

$$(f = 0)[x \mathbin{/\!\!/} \beta] \wedge (f' \geq 0)[x \mathbin{/\!\!/} \beta] \wedge (f'' \geq 0)[x \mathbin{/\!\!/} \beta]$$

---

$(g < 0)\big[x \mathbin{/\!\!/} (f, (1,1))\big]$:

$$(f > 0)[x \mathbin{/\!\!/} \beta]$$

$(g < 0)\big[x \mathbin{/\!\!/} (f, \{(2,1), (3,1), (4,1)\})\big]$:

$$(f > 0)[x \mathbin{/\!\!/} \beta] \vee (g < 0)[x \mathbin{/\!\!/} \alpha_1]$$

$(g < 0)\big[x \mathbin{/\!\!/} (f, \{(2,1), (3,2), (4,2)\})\big]$:

$$(g < 0)[x \mathbin{/\!\!/} \alpha_2] \vee ((f < 0)[x \mathbin{/\!\!/} \beta] \wedge (g < 0)[x \mathbin{/\!\!/} \alpha_1])$$

$(g < 0)\big[x \mathbin{/\!\!/} (f, \{(2,2), (3,2), (4,3)\})\big]$:

$$(f > 0)[x \mathbin{/\!\!/} \beta] \wedge (g < 0)[x \mathbin{/\!\!/} \alpha_2]$$

---

$(g \leq 0)\big[x \mathbin{/\!\!/} (f, (1,1))\big]$:

$$(f \geq 0)[x \mathbin{/\!\!/} \beta]$$

$(g \leq 0)\big[x \mathbin{/\!\!/} (f, \{(2,1), (3,1), (4,1)\})\big]$:

$$(f \geq 0)[x \mathbin{/\!\!/} \beta] \vee (g \leq 0)[x \mathbin{/\!\!/} \alpha_1]$$

$(g \leq 0)\big[x \mathbin{/\!\!/} (f, \{(2,1), (3,2), (4,2)\})\big]$:

$$(g \leq 0)[x \mathbin{/\!\!/} \alpha_2] \vee ((f \leq 0)[x \mathbin{/\!\!/} \beta] \wedge (g \leq 0)[x \mathbin{/\!\!/} \alpha_1])$$

$(g \leq 0)\big[x \mathbin{/\!\!/} (f, \{(2,2), (3,2), (4,3)\})\big]$:

$$(f \geq 0)[x \mathbin{/\!\!/} \beta] \wedge (g \leq 0)[x \mathbin{/\!\!/} \alpha_2]$$

**Target Atomic Formula of Degree Two**

Let $g = a^*x^2 + b^*x + c^*$, where $a^*$, $b^*$, $c^* \in \mathbb{Z}[\mathbf{u}]$. Assume that $a^* > 0$, and denote $D_g = {b^*}^2 - 4a^*c^*$, $\beta_1 = (g, (1,1))$, $\beta_2 = (g, (1,2))$. Then we have:

$(g = 0)\big[x /\!\!/ (f, (1,1))\big]$:

$$D_g \geq 0 \wedge ((f = 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2])$$

$(g = 0)\big[x /\!\!/ (f, \{(2,1), (3,1), (4,1)\})\big]$:

$$D_g \geq 0 \wedge \big((f = 0)[x /\!\!/ \beta_1] \wedge (f' \geq 0)[x /\!\!/ \beta_1] \wedge (f'' \leq 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2] \wedge (f' \geq 0)[x /\!\!/ \beta_2] \wedge (f'' \leq 0)[x /\!\!/ \beta_2]\big)$$

$(g = 0)\big[x /\!\!/ (f, \{(2,1), (3,2), (4,2)\})\big]$:

$$D_g \geq 0 \wedge \big((f = 0)[x /\!\!/ \beta_1] \wedge (f' \leq 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2] \wedge (f' \leq 0)[x /\!\!/ \beta_2]\big)$$

$(g = 0)\big[x /\!\!/ (f, \{(2,2), (3,2), (4,3)\})\big]$:

$$D_g \geq 0 \wedge \big((f = 0)[x /\!\!/ \beta_1] \wedge (f' \geq 0)[x /\!\!/ \beta_1] \wedge (f'' \geq 0)[x /\!\!/ \beta_1] \vee (f = 0)[x /\!\!/ \beta_2] \wedge (f' \geq 0)[x /\!\!/ \beta_2] \wedge (f'' \geq 0)[x /\!\!/ \beta_2]\big)$$

---

$(g < 0)\big[x /\!\!/ (f, (1,1))\big]$:

$$D_g > 0 \wedge (f < 0)[x /\!\!/ \beta_1] \wedge (f > 0)[x /\!\!/ \beta_2]$$

$(g < 0)\big[x /\!\!/ (f, \{(2,1), (3,1), (4,1)\})\big]$:

$$D_g > 0 \wedge \big((f < 0)[x /\!\!/ \beta_1] \wedge (f' \geq 0)[x /\!\!/ \beta_1] \wedge (f'' \leq 0)[x /\!\!/ \beta_1]\big) \wedge \big((f > 0)[x /\!\!/ \beta_2] \vee (g < 0)[x /\!\!/ \alpha_1]\big)$$

$(g < 0)\big[x /\!\!/ (f, \{(2,1), (3,2), (4,2)\})\big]$:

$$D_g > 0 \wedge \big((f > 0)[x /\!\!/ \beta_1] \vee (g < 0)[x /\!\!/ \alpha_1]\big) \wedge \big((f < 0)[x /\!\!/ \beta_2] \vee (g < 0)[x /\!\!/ \alpha_2]\big)$$

$(g < 0)\big[x /\!\!/ (f, \{(2,2), (3,2), (4,3)\})\big]$:

$$D_g > 0 \wedge \big((f < 0)[x /\!\!/ \beta_1] \vee (g < 0)[x /\!\!/ \alpha_2]\big) \wedge \big((f > 0)[x /\!\!/ \beta_2] \wedge (f' \geq 0)[x /\!\!/ \beta_2] \wedge (f'' \geq 0)[x /\!\!/ \beta_2]\big)$$

---

$(g \leq 0)\big[x /\!\!/ (f, (1,1))\big]$:

$$D_g \geq 0 \wedge (f \leq 0)[x /\!\!/ \beta_1] \wedge (f \geq 0)[x /\!\!/ \beta_2]$$

$(g \leq 0)\big[x /\!\!/ (f, \{(2,1), (3,1), (4,1)\})\big]$:

$$D_g \geq 0 \wedge \big((f \leq 0)[x /\!\!/ \beta_1] \wedge (f' \geq 0)[x /\!\!/ \beta_1] \wedge (f'' \leq 0)[x /\!\!/ \beta_1]\big) \wedge \big((f \geq 0)[x /\!\!/ \beta_2] \vee (g \leq 0)[x /\!\!/ \alpha_1]\big)$$

$(g \leq 0)\big[x /\!\!/ (f, \{(2,1),(3,2),(4,2)\})\big]$:

$$D_g \geq 0 \wedge \big((f \geq 0)[x /\!\!/ \beta_1] \wedge (f' \leq 0)[x /\!\!/ \beta_1] \vee (g \leq 0)[x /\!\!/ \alpha_1]\big) \wedge \big((f \leq 0)[x /\!\!/ \beta_2] \vee (g \leq 0)[x /\!\!/ \alpha_2]\big)$$

$(g \leq 0)\big[x /\!\!/ (f, \{(2,2),(3,2),(4,3)\})\big]$:

$$D_g \geq 0 \wedge \big((f \leq 0)[x /\!\!/ \beta_1] \vee (g \leq 0)[x /\!\!/ \alpha_2]\big) \wedge \big((f \geq 0)[x /\!\!/ \beta_2] \wedge (f' \geq 0)[x /\!\!/ \beta_2] \wedge (f'' \geq 0)[x /\!\!/ \beta_2]\big)$$

# Appendix B

# Results of the Computational Experiments

In this appendix we list detailed results of our computational experiments discussed in Section 6.3. There we have considered three real quantifier elimination example sets referred to as "Bath," "Regressions," and "Remis." The example sets contain 51, 19, and 154 examples, respectively.

The results are listed in Tables B.1–B.12. The tables have the same form and list one example per line. Each line contains the following information about the example: name, input formula information, output formula information, and the computation time. We consider three real QE implementations: (1) the Redlog's original implementation of the quadratic quantifier elimination, (2) our implementation without clustering, and (3) our implementation with clustering.

Each example is a Tarski formula in the prenex normal form. Information on the input formula are shown in columns "#q," "#p," and "#at" that list the number of quantified variables, the number of parameters, and the number of atomic formulas, respectively. For the sake of a concise description, we do not list the number of quantifier alternations or the types of quantifier prefixes.

Information on computed equivalents of the input formula is shown in column triples "Original," "Without Clustering," and "With Clustering." Each column triple consists of columns "#q," "#at," and "Time" that list the number of quantified variables, the number of atomic formulas, and the computation time of the respective implementation, respectively.

Observe that a zero in the input column "#p" means that the example is actually a decision problem. A nonzero number of quantified variables on output means that an implementation was not able to eliminate all quantifiers.

Finally, the number of atomic formulas in an output formula can be zero, which means that either "true" or "false" was obtained. The reason for this is that we count only nontrivial atomic formulas, i.e., "true" and "false" do not count as atomic formulas, and do not occur in nontrivial output formulas thank to simplification.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| 01-parabola | 1 | 3 | 1 | 0 | 7 | <10 ms | 0 | 15 | <10 ms | 0 | 7 | <10 ms |
| 02-whitney | 2 | 3 | 3 | 0 | 5 | <10 ms | 0 | 8 | <10 ms | 0 | 6 | <10 ms |
| 03-quartic | 1 | 3 | 1 | 1 | 1 | 10 ms | 1 | 1 | <10 ms | 1 | 1 | <10 ms |
| 07-rimpl | 2 | 3 | 3 | 1 | 11 | <10 ms | 0 | 30 | 10 ms | 0 | 27 | <10 ms |
| 08-ball | 3 | 0 | 2 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | <10 ms |
| 09-trw | 3 | 0 | 3 | 0 | 0 | 70 ms | 0 | 0 | 10 ms | 0 | 0 | 20 ms |
| 10-coljohn | 1 | 2 | 6 | 0 | 27 | 10 ms | 0 | 24 | <10 ms | 0 | 24 | <10 ms |
| 11-lbounds | 5 | 1 | 3 | 0 | 1 | <10 ms | 0 | 1 | 10 ms | 0 | 1 | <10 ms |
| 12-xellipse | 2 | 3 | 3 | 0 | 52 | 20 ms | 0 | 60 | 20 ms | 0 | 39 | 20 ms |
| 13-davhein | 3 | 1 | 5 | 0 | 3 | 10 ms | 0 | 3 | <10 ms | 0 | 3 | <10 ms |
| 14-hong-90 | 2 | 3 | 3 | 0 | 1 | 10 ms | 0 | 1 | <10 ms | 0 | 1 | <10 ms |
| 15-solotareff-3 | 2 | 3 | 8 | 0 | 9 | 10 ms | 0 | 9 | <10 ms | 0 | 9 | <10 ms |
| 16-collision | 3 | 0 | 7 | 0 | 0 | 10 ms | 0 | 0 | <10 ms | 0 | 0 | 10 ms |
| 17-trivariate | 3 | 0 | 1 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| 18-ellipse | 2 | 4 | 3 | 1 | 11 | 10 ms | 1 | 14 | 40 ms | 1 | 12 | 20 ms |
| 23-piano | 2 | 2 | 9 | 0 | 468 | 150 ms | 0 | 320 | 80 ms | 0 | 408 | 99 ms |
| 24-int-a | 1 | 2 | 3 | 0 | 7 | <10 ms | 0 | 10 | <10 ms | 0 | 8 | <10 ms |

Table B.1: Bath example set results Part I.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| 25-int-a-g | 1 | 2 | 4 | 0 | 19 | 10 ms | 0 | 28 | 10 ms | 0 | 20 | 10 ms |
| 26-int-b | 1 | 2 | 3 | 0 | 7 | 10 ms | 0 | 10 | <10 ms | 0 | 8 | <10 ms |
| 27-int-b-g | 1 | 2 | 4 | 0 | 19 | 10 ms | 0 | 18 | <10 ms | 0 | 20 | <10 ms |
| 28-rand-a | 3 | 0 | 3 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| 29-rand-a-g | 3 | 0 | 3 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| 30-rand-b | 3 | 0 | 3 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| 31-rand-b-g | 3 | 0 | 3 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| 32-ellipse-a | 2 | 3 | 8 | 0 | 43 | 10 ms | 0 | 81 | 10 ms | 0 | 43 | 10 ms |
| 33-ellipse-a-g | 2 | 3 | 8 | 0 | 43 | 10 ms | 0 | 81 | 20 ms | 0 | 43 | 10 ms |
| 34-ellipse-b | 2 | 3 | 8 | 0 | 43 | 10 ms | 0 | 81 | 10 ms | 0 | 43 | 10 ms |
| 35-ellipse-b-g | 2 | 3 | 11 | 0 | 43 | 10 ms | 0 | 81 | 30 ms | 0 | 43 | 20 ms |
| 36-solotareff-a | 2 | 2 | 12 | 0 | 2 | <10 ms | 0 | 2 | <10 ms | 0 | 2 | <10 ms |
| 37-solotareff-a-g | 2 | 2 | 12 | 0 | 2 | <10 ms | 0 | 2 | <10 ms | 0 | 2 | <10 ms |
| 38-solotareff-b | 2 | 2 | 12 | 0 | 2 | <10 ms | 0 | 2 | 10 ms | 0 | 2 | 10 ms |
| 39-solotareff-b-g | 2 | 2 | 12 | 0 | 2 | 10 ms | 0 | 2 | 10 ms | 0 | 2 | <10 ms |
| 40-collision-a | 3 | 1 | 4 | 1 | 102 | 140 ms | 1 | 192 | 420 ms | 1 | 184 | 440 ms |
| 41-collision-a-g | 3 | 1 | 7 | 1 | 2377 | 8 s | 1 | 6412 | 17 s | 1 | 3268 | 9 s |

Table B.2: Bath example set results Part II.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| 42-collision-b | 3 | 1 | 4 | 2 | 11 | 20 ms | 2 | 32 | 30 ms | 2 | 12 | 30 ms |
| 43-collision-b-g | 3 | 2 | 8 | 2 | 21 | 190 ms | 2 | 40 | 340 ms | 2 | 22 | 310 ms |
| 44-off-center | 2 | 1 | 3 | 1 | 4 | 10 ms | 1 | 9 | 10 ms | 1 | 7 | 10 ms |
| 47-edges | 3 | 2 | 12 | 0 | 92 | 10 ms | 0 | 103 | 10 ms | 0 | 90 | 10 ms |
| 48-simpl-edges | 1 | 2 | 11 | 0 | 235 | 30 ms | 0 | 271 | 40 ms | 0 | 239 | 30 ms |
| 49-putnum | 4 | 2 | 4 | 0 | 53 | 30 ms | 0 | 114 | 90 ms | 0 | 80 | 70 ms |
| 50-simpl-putnum | 2 | 2 | 2 | 0 | 53 | 30 ms | 0 | 98 | 90 ms | 0 | 62 | 60 ms |
| 51-yang-xia | 3 | 3 | 10 | 1 | 7 | <10 ms | 1 | 7 | 10 ms | 1 | 7 | 10 ms |
| 52-simpl-yang-xia | 1 | 3 | 9 | 1 | 7 | <10 ms | 1 | 7 | 10 ms | 1 | 7 | <10 ms |
| 53-seit | 4 | 7 | 15 | 0 | 19 | 40 ms | 0 | 29 | 20 ms | 0 | 19 | 30 ms |
| 54-simpl-seit | 1 | 7 | 16 | 0 | 32 | 129 ms | 0 | 41 | 210 ms | 0 | 30 | 139 ms |
| 55-cyclic-3 | 2 | 1 | 3 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| 56-cyclic-4 | 3 | 1 | 4 | 0 | 1 | <10 ms | 0 | 1 | <10 ms | 0 | 1 | 10 ms |
| 57-joukowsky-1 | 4 | 0 | 6 | 4 | 6 | <10 ms | 3 | 550 | 2 s | 2 | 345 | 1 s |
| 58-joukowsky-2 | 4 | 0 | 6 | 4 | 6 | 10 ms | 3 | 550 | 2 s | 2 | 345 | 1 s |
| 59-separate | 4 | 0 | 5 | 4 | 5 | 10 ms | 3 | 251 | 2 s | 2 | 149 | 1 s |
| 60-uhp | 4 | 0 | 6 | 4 | 6 | <10 ms | 3 | 371 | 1 s | 2 | 233 | 900 ms |

Table B.3: Bath example set results Part III.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| ab07hopf9 | 9 | 0 | 13 | 2 | 2463 | 2 min | 2 | 21965 | 29 min | 2 | 30143 | 18 min |
| aci | 2 | 8 | 4 | 0 | 29 | 10 ms | 0 | 59 | 10 ms | 0 | 35 | <10 ms |
| as5v | 5 | 0 | 14 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| ccon | 3 | 0 | 2 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| ell | 2 | 3 | 2 | 0 | 52 | 20 ms | 0 | 60 | 20 ms | 0 | 39 | 20 ms |
| gatermann | 10 | 0 | 15 | 0 | 0 | 19 s | 0 | 0 | 2 s | 0 | 0 | 17 s |
| hong | 2 | 2 | 2 | 1 | 23 | <10 ms | 1 | 28 | 10 ms | 1 | 19 | 10 ms |
| mc4 | 3 | 0 | 3 | 0 | 0 | 40 ms | 0 | 0 | 30 ms | 0 | 0 | 30 ms |
| mtp2 | 9 | 6 | 15 | 0 | 84 | 10 ms | 0 | 84 | 20 ms | 0 | 84 | 10 ms |
| mtp3 | 27 | 27 | 54 | 0 | 1456785 | 15 min | 0 | 1451196 | 54 min | 0 | 1451196 | 53 min |
| p9 | 11 | 0 | 38 | 0 | 0 | 109 ms | 0 | 0 | 210 ms | 0 | 0 | 200 ms |
| quartic | 1 | 3 | 1 | 1 | 1 | <10 ms | 1 | 1 | <10 ms | 1 | 1 | <10 ms |
| rp2 | 7 | 3 | 56 | 0 | 135 | 80 ms | 0 | 60 | 129 ms | 0 | 60 | 129 ms |
| rp3 | 10 | 2 | 128 | 0 | 769 | 44 s | 0 | 94 | 47 s | 0 | 94 | 48 s |
| sl9 | 7 | 2 | 13 | 1 | 315 | 360 ms | 2 | 2491 | 9 s | 0 | 1370 | 18 s |
| snip | 3 | 0 | 4 | 0 | 0 | 10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| solo2 | 2 | 2 | 12 | 0 | 2 | <10 ms | 0 | 2 | 10 ms | 0 | 2 | <10 ms |
| wilk | 1 | 0 | 3 | 1 | 3 | 10 ms | 1 | 3 | 20 ms | 1 | 3 | 30 ms |
| wnip | 3 | 0 | 4 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |

Table B.4: Regressions example set results.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| acines | 2 | 8 | 4 | 0 | 29 | <10 ms | 0 | 59 | 10 ms | 0 | 35 | 10 ms |
| clo-1 | 2 | 3 | 3 | 0 | 8 | <10 ms | 0 | 14 | <10 ms | 0 | 10 | <10 ms |
| clo-2 | 2 | 3 | 3 | 1 | 11 | <10 ms | 0 | 30 | <10 ms | 0 | 27 | <10 ms |
| clo-3 | 2 | 3 | 3 | 0 | 5 | <10 ms | 0 | 8 | <10 ms | 0 | 6 | <10 ms |
| clo-4 | 2 | 3 | 5 | 0 | 3 | <10 ms | 0 | 6 | <10 ms | 0 | 3 | <10 ms |
| clo-5 | 1 | 2 | 4 | 1 | 6 | <10 ms | 0 | 98 | 50 ms | 0 | 60 | 30 ms |
| clo-6 | 1 | 9 | 3 | 0 | 34 | 220 ms | 0 | 109 | 510 ms | 0 | 46 | 730 ms |
| coljoh-1 | 1 | 2 | 6 | 0 | 6 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | <10 ms |
| coljoh-2 | 1 | 2 | 4 | 0 | 20 | 10 ms | 0 | 19 | <10 ms | 0 | 19 | <10 ms |
| collision-1 | 3 | 0 | 7 | 0 | 0 | 10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| collision-2 | 3 | 0 | 7 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| consistency | 3 | 0 | 2 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | <10 ms |
| davhei | 3 | 1 | 5 | 0 | 3 | 10 ms | 0 | 3 | <10 ms | 0 | 3 | <10 ms |
| dsw96_ex01 | 5 | 3 | 6 | 0 | 3 | 10 ms | 0 | 19 | 30 ms | 0 | 7 | 10 ms |
| dsw96_ex02 | 3 | 2 | 6 | 0 | 47 | 10 ms | 0 | 10 | 10 ms | 0 | 10 | 10 ms |
| dsw96_ex03 | 6 | 0 | 7 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| dsw96_ex04 | 3 | 4 | 4 | 0 | 26 | 10 ms | 0 | 68 | 10 ms | 0 | 34 | 10 ms |
| dsw96_ex05 | 7 | 5 | 8 | n/a | n/a | >1 h | n/a | n/a | >1 h | n/a | n/a | >1 h |
| dsw96_ex06 | 4 | 3 | 5 | 0 | 4 | <10 ms | 0 | 10 | 30 ms | 0 | 4 | 20 ms |
| dsw96_ex07 | 7 | 4 | 8 | n/a | n/a | >1 h | 0 | 5088 | 2 s | 0 | 279 | 200 ms |

Table B.5: Remis example set results Part I.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| dsw96_ex08 | 2 | 0 | 3 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| dsw96_ex09 | 3 | 1 | 4 | 1 | 90 | 129 ms | 0 | 1 | 20 ms | 0 | 1 | 10 ms |
| dsw96_ex10 | 9 | 3 | 10 | n/a | n/a | >1 h | 1 | 20497 | 44 s | 0 | 815 | 6 s |
| dsw96_ex11 | 4 | 3 | 6 | 0 | 0 | 10 ms | 0 | 0 | <10 ms | 0 | 0 | 10 ms |
| dsw96_ex12 | 7 | 4 | 8 | 1 | 1280 | 610 ms | n/a | n/a | 1 min | 0 | 2084 | 4 s |
| dsw96_ex13 | 4 | 6 | 5 | 0 | 41 | 20 ms | 0 | 284 | 90 ms | 0 | 55 | 40 ms |
| dsw96_ex14 | 4 | 5 | 5 | 0 | 58 | 10 ms | 0 | 446 | 119 ms | 0 | 78 | 50 ms |
| ellipse | 2 | 3 | 2 | 0 | 52 | 20 ms | 0 | 60 | 30 ms | 0 | 39 | 20 ms |
| ex1_square-1 | 2 | 2 | 11 | 0 | 10 | <10 ms | 0 | 18 | <10 ms | 0 | 18 | 10 ms |
| ex1_square-2 | 4 | 4 | 17 | 0 | 29 | <10 ms | 0 | 87 | 10 ms | 0 | 87 | 10 ms |
| ex1_square-3 | 4 | 4 | 17 | 0 | 25 | <10 ms | 0 | 83 | 10 ms | 0 | 83 | 10 ms |
| ex1_square-4 | 28 | 0 | 102 | 0 | 0 | 700 ms | 0 | 0 | 940 ms | 0 | 0 | 920 ms |
| ex1_square_path-1 | 2 | 2 | 11 | 0 | 10 | <10 ms | 0 | 18 | <10 ms | 0 | 18 | 10 ms |
| ex1_square_path-2 | 4 | 4 | 17 | 0 | 29 | <10 ms | 0 | 87 | 10 ms | 0 | 87 | 10 ms |
| ex1_square_path-3 | 4 | 4 | 17 | 0 | 25 | <10 ms | 0 | 83 | 10 ms | 0 | 83 | 10 ms |
| ex1_square_path-4 | 28 | 1 | 103 | 0 | 1 | 3 s | 0 | 1 | 19 s | 0 | 1 | 19 s |
| ex1_square_path-5 | 28 | 0 | 103 | 0 | 0 | 740 ms | 0 | 0 | 960 ms | 0 | 0 | 1 s |
| ex2_lshape-1 | 2 | 2 | 16 | 0 | 10 | <10 ms | 0 | 16 | 10 ms | 0 | 16 | <10 ms |
| ex2_lshape-2 | 4 | 4 | 22 | 0 | 27 | 10 ms | 0 | 129 | 20 ms | 0 | 129 | 20 ms |
| ex2_lshape-3 | 4 | 4 | 22 | 0 | 24 | 10 ms | 0 | 129 | 20 ms | 0 | 129 | 20 ms |

Table B.6: Remis example set results Part II.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| ex2_lshape-4 | 28 | 0 | 132 | 0 | 0 | 4 s | 0 | 0 | 3 s | 0 | 0 | 3 s |
| ex3_direction-1 | 2 | 2 | 12 | 0 | 28 | 10 ms | 0 | 31 | <10 ms | 0 | 31 | 10 ms |
| ex3_direction-2 | 4 | 4 | 18 | 0 | 91 | 10 ms | 0 | 147 | 20 ms | 0 | 147 | 20 ms |
| ex3_direction-3 | 4 | 4 | 18 | 0 | 105 | 10 ms | 0 | 167 | 20 ms | 0 | 167 | 30 ms |
| ex3_direction-4 | 28 | 0 | 108 | 0 | 0 | 6 s | 0 | 0 | 4 s | 0 | 0 | 4 s |
| ex3_direction-5 | 28 | 1 | 109 | 0 | 1 | 7 s | 0 | 1 | 10 min | 0 | 1 | 10 min |
| ex3_direction-6 | 28 | 0 | 109 | 0 | 0 | 6 s | 0 | 0 | 4 s | 0 | 0 | 4 s |
| ex4_walls-1 | 2 | 4 | 27 | 0 | 59 | 10 ms | 0 | 66 | 10 ms | 0 | 66 | <10 ms |
| ex4_walls-2 | 2 | 4 | 27 | 0 | 93 | 10 ms | 0 | 82 | 10 ms | 0 | 82 | 10 ms |
| ex4_walls-3 | 22 | 0 | 162 | 0 | 0 | 3 s | 0 | 0 | 3 s | 0 | 0 | 3 s |
| ex5_oblique-1 | 3 | 3 | 9 | 0 | 6 | <10 ms | 0 | 6 | <10 ms | 0 | 6 | <10 ms |
| ex5_oblique-2 | 5 | 6 | 16 | 0 | 20 | 10 ms | 0 | 21 | 10 ms | 0 | 21 | <10 ms |
| ex5_oblique-3 | 5 | 6 | 16 | 0 | 9 | 10 ms | 0 | 16 | 10 ms | 0 | 16 | 10 ms |
| ex5_oblique-4 | 5 | 6 | 16 | 0 | 20 | <10 ms | 0 | 29 | 10 ms | 0 | 29 | 10 ms |
| ex5_oblique-5 | 51 | 0 | 144 | 0 | 0 | 1 s | 0 | 0 | 930 ms | 0 | 0 | 950 ms |
| ex6_cube-1 | 3 | 3 | 18 | 0 | 45 | <10 ms | 0 | 38 | 10 ms | 0 | 38 | <10 ms |
| ex6_cube-2 | 5 | 6 | 25 | 0 | 156 | 20 ms | 0 | 160 | 30 ms | 0 | 160 | 20 ms |
| ex6_cube-3 | 5 | 6 | 25 | 0 | 165 | 20 ms | 0 | 166 | 20 ms | 0 | 166 | 30 ms |
| ex6_cube-4 | 5 | 6 | 25 | 0 | 84 | 10 ms | 0 | 137 | 20 ms | 0 | 137 | 20 ms |

Table B.7: Remis example set results Part III.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| ex6_cube-5 | 51 | 0 | 225 | n/a | n/a | >1 h | 0 | 0 | 30 min | 0 | 0 | 24 min |
| ex7_lshape_obst-1 | 2 | 2 | 22 | 0 | 52 | 10 ms | 0 | 46 | 10 ms | 0 | 46 | 10 ms |
| ex7_lshape_obst-2 | 4 | 4 | 28 | 0 | 165 | 20 ms | 0 | 269 | 30 ms | 0 | 269 | 30 ms |
| ex7_lshape_obst-3 | 4 | 4 | 28 | 0 | 165 | 20 ms | 0 | 245 | 30 ms | 0 | 245 | 20 ms |
| ex7_lshape_obst-4 | 4 | 4 | 28 | 0 | 279 | 20 ms | 0 | 339 | 30 ms | 0 | 339 | 40 ms |
| ex7_lshape_obst-5 | 54 | 0 | 336 | n/a | n/a | >1 h | n/a | n/a | >1 h | n/a | n/a | >1 h |
| ex7_lshape_obst-6 | 68 | 0 | 420 | n/a | n/a | >1 h | n/a | n/a | >1 h | n/a | n/a | >1 h |
| ex8_oblique_obst-1 | 2 | 2 | 20 | 0 | 74 | <10 ms | 0 | 62 | 10 ms | 0 | 62 | 10 ms |
| ex8_oblique_obst-2 | 4 | 4 | 26 | 0 | 281 | 20 ms | 0 | 311 | 40 ms | 0 | 311 | 30 ms |
| ex8_oblique_obst-3 | 4 | 4 | 26 | 0 | 256 | 20 ms | 0 | 308 | 40 ms | 0 | 308 | 40 ms |
| ex8_oblique_obst-4 | 4 | 4 | 26 | 0 | 269 | 30 ms | 0 | 330 | 30 ms | 0 | 330 | 40 ms |
| ex8_oblique_obst-5 | 54 | 0 | 312 | n/a | n/a | >1 h | n/a | n/a | >1 h | n/a | n/a | >1 h |
| ex9_variant2_ex8-1 | 2 | 2 | 20 | 0 | 74 | <10 ms | 0 | 62 | 10 ms | 0 | 62 | 10 ms |
| ex9_variant2_ex8-2 | 2 | 3 | 20 | 0 | 74 | <10 ms | 0 | 62 | 10 ms | 0 | 62 | 10 ms |
| ex9_variant2_ex8-3 | 2 | 3 | 20 | 0 | 74 | 10 ms | 0 | 62 | 10 ms | 0 | 62 | 10 ms |
| ex9_variant2_ex8-4 | 2 | 3 | 20 | 0 | 74 | 10 ms | 0 | 62 | 10 ms | 0 | 62 | <10 ms |
| ex9_variant2_ex8-5 | 26 | 0 | 156 | 0 | 0 | 38 s | 0 | 0 | 2 min | 0 | 0 | 2 min |
| ex9_variant_ex8-1 | 2 | 2 | 20 | 0 | 74 | <10 ms | 0 | 62 | 10 ms | 0 | 62 | 10 ms |
| ex9_variant_ex8-2 | 4 | 4 | 26 | 0 | 281 | 20 ms | 0 | 311 | 40 ms | 0 | 311 | 30 ms |

Table B.8: Remis example set results Part IV.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| ex9_variant_ex8-3 | 4 | 4 | 26 | 0 | 256 | 20 ms | 0 | 308 | 40 ms | 0 | 308 | 30 ms |
| ex9_variant_ex8-4 | 4 | 4 | 26 | 0 | 269 | 30 ms | 0 | 330 | 40 ms | 0 | 330 | 40 ms |
| ex9_variant_ex8-5 | 40 | 0 | 234 | n/a | n/a | >1 h | n/a | n/a | >1 h | n/a | n/a | >1 h |
| gen5 | 1 | 15 | 8 | 0 | 393 | 99 ms | 0 | 563 | 180 ms | 0 | 283 | 80 ms |
| hongsbet | 2 | 2 | 2 | 1 | 23 | 10 ms | 1 | 28 | 10 ms | 1 | 19 | 10 ms |
| hong | 1 | 3 | 3 | 0 | 18 | <10 ms | 0 | 48 | 10 ms | 0 | 20 | 10 ms |
| linequadbe | 1 | 3 | 3 | 0 | 25 | <10 ms | 0 | 18 | 10 ms | 0 | 16 | 10 ms |
| mccallum-01 | 3 | 0 | 2 | 0 | 0 | 20 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| mccallum-02 | 3 | 0 | 2 | 0 | 0 | 10 ms | 0 | 0 | 10 ms | 0 | 0 | <10 ms |
| mccallum-03 | 3 | 0 | 3 | 0 | 0 | 10 ms | 0 | 0 | <10 ms | 0 | 0 | <10 ms |
| mccallum-04 | 3 | 0 | 3 | 0 | 0 | 50 ms | 0 | 0 | 20 ms | 0 | 0 | 20 ms |
| mccallum-05 | 3 | 0 | 4 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | 10 ms |
| mccallum-06 | 3 | 0 | 4 | 0 | 0 | 10 ms | 0 | 0 | <10 ms | 0 | 0 | 10 ms |
| mccallum-07 | 3 | 0 | 4 | 1 | 302 | 190 ms | 1 | 270 | 190 ms | 1 | 241 | 109 ms |
| mccallum-08 | 3 | 0 | 2 | 3 | 2 | 10 ms | 3 | 2 | <10 ms | 3 | 2 | <10 ms |
| mccallum-09 | 3 | 0 | 6 | 0 | 0 | 270 ms | 0 | 0 | 109 ms | 0 | 0 | 99 ms |
| mccallum-10 | 3 | 0 | 6 | 3 | 5 | <10 ms | 3 | 5 | <10 ms | 3 | 5 | <10 ms |
| mccallum-11 | 3 | 0 | 5 | 3 | 5 | <10 ms | 3 | 5 | 10 ms | 3 | 5 | <10 ms |
| mccallum-12 | 3 | 0 | 7 | 3 | 7 | <10 ms | 3 | 7 | <10 ms | 3 | 7 | <10 ms |

Table B.9: Remis example set results Part V.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| motzkin | 2 | 0 | 1 | 0 | 0 | <10 ms | 0 | 0 | <10 ms | 0 | 0 | 10 ms |
| mtp2 | 9 | 6 | 15 | 0 | 84 | 10 ms | 0 | 84 | 10 ms | 0 | 84 | 20 ms |
| mtp3 | 27 | 27 | 54 | 0 | 1456785 | 15 min | 0 | 1451196 | 53 min | 0 | 1451196 | 53 min |
| netan1-01 | 8 | 0 | 14 | 0 | 0 | 10 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| netan1-02 | 8 | 0 | 14 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | <10 ms |
| netan1-03 | 18 | 0 | 31 | 0 | 0 | 10 ms | 0 | 0 | 30 ms | 0 | 0 | 30 ms |
| netan1-04 | 8 | 1 | 14 | 0 | 1 | 10 ms | 0 | 1 | 10 ms | 0 | 1 | 10 ms |
| netan1-05 | 9 | 0 | 14 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| netan1-06 | 9 | 0 | 14 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| netan1-07 | 9 | 0 | 14 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| netan1-08 | 9 | 0 | 14 | 0 | 0 | 30 ms | 0 | 0 | 20 ms | 0 | 0 | 10 ms |
| netan1-09 | 13 | 5 | 18 | 0 | 650 | 2 s | 0 | 10 | 20 ms | 0 | 5 | 10 ms |
| netan1-10 | 17 | 0 | 18 | 0 | 0 | <10 ms | 0 | 0 | 20 ms | 0 | 0 | 20 ms |
| netan1-11 | 17 | 0 | 18 | 0 | 0 | 10 ms | 0 | 0 | 10 ms | 0 | 0 | 20 ms |
| netan2-1 | 8 | 6 | 9 | 0 | 128 | 10 ms | 0 | 409 | 99 ms | 0 | 107 | 40 ms |
| netan2-2 | 8 | 2 | 9 | 0 | 2 | <10 ms | 0 | 2 | 10 ms | 0 | 4 | 10 ms |
| period9 | 11 | 0 | 38 | 0 | 0 | 99 ms | 0 | 0 | 220 ms | 0 | 0 | 200 ms |
| prob1 | 2 | 1 | 1 | 1 | 1 | <10 ms | 1 | 1 | <10 ms | 1 | 1 | <10 ms |
| prob2-1 | 3 | 1 | 4 | 0 | 188 | 90 ms | 0 | 67 | 40 ms | 0 | 82 | 40 ms |

Table B.10: Remis example set results Part VI.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| prob2-2 | 3 | 2 | 4 | 3 | 4 | <10 ms | 1 | 134006 | 10 min | 1 | 185705 | 10 min |
| prob3 | 3 | 0 | 4 | 3 | 4 | 10 ms | 3 | 7 | 10 ms | 3 | 7 | 10 ms |
| prob4 | 4 | 0 | 9 | 4 | 9 | 10 ms | 3 | 659 | 20 s | 3 | 473 | 14 s |
| progruntimeprop | 8 | 0 | 8 | 0 | 0 | 10 ms | 0 | 0 | 10 ms | 0 | 0 | 10 ms |
| scheduling | 11 | 2 | 37 | 0 | 58 | 190 ms | 0 | 68 | 1 s | 0 | 68 | 1 s |
| sturm99a_ex3-22 | 7 | 2 | 13 | 1 | 315 | 370 ms | 2 | 2491 | 9 s | 0 | 1370 | 18 s |
| sw97_ex01 | 4 | 9 | 5 | 0 | 12 | 10 ms | 0 | 148 | 90 ms | 0 | 76 | 40 ms |
| sw97_ex02 | 4 | 9 | 6 | 0 | 20 | 10 ms | 0 | 136 | 109 ms | 0 | 82 | 60 ms |
| sw97_ex03 | 4 | 9 | 6 | 0 | 11 | <10 ms | 0 | 112 | 70 ms | 0 | 70 | 50 ms |
| sw97_ex04 | 4 | 9 | 6 | 0 | 5 | 10 ms | 0 | 160 | 89 ms | 0 | 58 | 50 ms |
| sw97_ex05 | 4 | 9 | 5 | 0 | 12 | 10 ms | 0 | 142 | 90 ms | 0 | 49 | 30 ms |
| sw97_ex06 | 4 | 10 | 6 | 0 | 48 | 370 ms | 0 | 103 | 1 s | 0 | 43 | 700 ms |
| sw97_ex07 | 35 | 0 | 58 | n/a | n/a | >1 h | n/a | n/a | >1 h | n/a | n/a | >1 h |
| sw97_ex08 | 3 | 11 | 4 | 0 | 460 | 80 ms | 0 | 2008 | 630 ms | 0 | 459 | 160 ms |
| sw97_ex09 | 3 | 6 | 3 | 0 | 2954 | 6 min | 0 | 4758 | 46 s | 0 | 3380 | 37 s |
| sw97_ex10 | 43 | 0 | 85 | n/a | n/a | >1 h | 29 | 2584 | 5 s | 29 | 482 | 3 s |
| sw97_ex11 | 1 | 5 | 2 | 0 | 5 | <10 ms | 0 | 9 | 10 ms | 0 | 6 | <10 ms |
| sw97_ex12-1 | 10 | 5 | 24 | 0 | 544 | 70 ms | 0 | 440 | 129 ms | 0 | 440 | 129 ms |
| sw97_ex12-2 | 4 | 4 | 6 | 0 | 25 | 10 ms | 0 | 37 | 20 ms | 0 | 55 | 20 ms |

Table B.11: Remis example set results Part VII.

| | Input | | | Original | | | Without Clustering | | | With Clustering | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #q | #p | #at | #q | #at | Time | #q | #at | Time | #q | #at | Time |
| sw97_ex13 | 3 | 4 | 4 | 1 | 2 | <10 ms | 0 | 96 | 109 ms | 0 | 72 | 90 ms |
| sw97_ex14-1 | 2 | 3 | 5 | 0 | 68 | <10 ms | 0 | 16 | <10 ms | 0 | 18 | 10 ms |
| sw97_ex14-2 | 2 | 3 | 5 | 0 | 86 | 10 ms | 0 | 60 | 10 ms | 0 | 37 | 10 ms |
| sw97_ex15 | 1 | 2 | 9 | 0 | 8 | <10 ms | 0 | 27 | <10 ms | 0 | 27 | <10 ms |
| sw97_ex16 | 3 | 2 | 11 | 0 | 1408 | 500 ms | 0 | 920 | 440 ms | 0 | 1264 | 490 ms |
| sw97_ex17 | 3 | 4 | 14 | 0 | 33137 | 4 min | 0 | 572 | 4 s | 0 | 22608 | 5 min |
| sw97_ex18 | 1 | 3 | 2 | 0 | 1 | <10 ms | 0 | 1 | <10 ms | 0 | 1 | <10 ms |
| sw97_ex19 | 3 | 2 | 6 | 0 | 1042 | 310 ms | 0 | 352 | 109 ms | 0 | 232 | 150 ms |
| sw97_ex20-1 | 4 | 9 | 8 | 0 | 34432 | 2 min | 0 | 107025 | 3 min | 0 | 26261 | 3 min |
| sw97_ex20-2 | 4 | 3 | 8 | 0 | 12313 | 6 s | 0 | 6679 | 2 s | 0 | 1792 | 2 s |
| sw97_ex20-3 | 7 | 0 | 9 | 0 | 0 | <10 ms | 0 | 0 | 10 ms | 0 | 0 | 20 ms |
| sw97_ex21-1 | 1 | 10 | 81 | 0 | 2995 | 230 ms | 0 | 2661 | 210 ms | 0 | 2106 | 220 ms |
| sw97_ex21-2 | 1 | 12 | 98 | 0 | 4545 | 380 ms | 0 | 3324 | 330 ms | 0 | 3324 | 330 ms |
| sw97_ex21-3 | 1 | 2 | 81 | 0 | 41 | <10 ms | 0 | 8 | 10 ms | 0 | 41 | <10 ms |
| sw97_ex21-4 | 1 | 2 | 98 | 0 | 148 | 20 ms | 0 | 33 | 10 ms | 0 | 33 | 10 ms |
| sw97_ex21-5 | 3 | 0 | 101 | 0 | 0 | 10 ms | 0 | 0 | <10 ms | 0 | 0 | 10 ms |
| sw97_ex21-6 | 3 | 0 | 101 | 0 | 0 | 10 ms | 0 | 0 | 20 ms | 0 | 0 | 20 ms |
| termination | 3 | 0 | 3 | 0 | 0 | 70 ms | 0 | 0 | 10 ms | 0 | 0 | 30 ms |
| triangquadbe | 2 | 6 | 4 | 0 | 598 | 38 s | 0 | 653 | 6 s | 0 | 362 | 54 s |

Table B.12: Remis example set results Part VIII.

# Bibliography


[1] T. Achterberg. *Constraint Integer Programming*. Doctoral dissertation, Technische Universität Berlin, Germany, 2007.

[2] E. A. Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM Journal on Computing*, 2(1):1–6, 1973.

[3] A. G. Akritas. Linear and quadratic complexity bounds on the values of the positive roots of polynomials. *Journal of Universal Computer Science*, 15(3):523–537, 2009.

[4] R. Bar-Yehuda and S. Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.

[5] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. In *Analysis and Design of Algorithms for Combinatorial Problems*, volume 109 of *North-Holland Mathematics Studies*, pages 27–45. North-Holland, 1985.

[6] S. Basu, R. Pollack, and M.-F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM*, 43(6):1002–1045, 1996.

[7] M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. *Journal of Computer and System Sciences*, 32(2):251–264, 1986.

[8] J. Bochnak, M. Coste, and M.-F. Roy. *Real Algebraic Geometry*. Springer, 1998.

[9] F. Boulier, M. Lefranc, F. Lemaire, P.-E. Morant, and A. Ürgüplü. On proving the absence of oscillations in models of genetic circuits. In *Proceedings of Algebraic Biology 2007*, volume 4545 of *LNCS*, pages 66–80. Springer, 2007.

[10] R. Bradford, J. H. Davenport, M. England, S. McCallum, and D. Wilson. Truth table invariant cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 76:1–35, 2016.

[11] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[12] C. W. Brown. *Solution Formula Construction for Truth Invariant CAD's.* Doctoral dissertation, University of Delaware, USA, 1999.

[13] C. W. Brown. QEPCAD B: A program for computing with semi-algebraic sets using CADs. *SIGSAM Bulletin*, 37(4):97–108, 2003.

[14] C. W. Brown. Constructing a single open cell in a cylindrical algebraic decomposition. In *Proceedings of the ISSAC 2013*, pages 133–140. ACM, 2013.

[15] C. W. Brown, M. El Kahoui, D. Novotni, and A. Weber. Algorithmic methods for investigating equilibria in epidemic modeling. *Journal of Symbolic Computation*, 41(11):1157–1173, 2006.

[16] C. W. Brown and M. Košta. Constructing a single cell in cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 70:14–48, 2015.

[17] K.-D. Burhenne. Implementierung eines Algorithmus zur Quantorenelimination fur lineare reelle Probleme. Master's thesis, Universität Passau, Germany, 1990.

[18] C. Chen and M. M. Maza. Quantifier elimination by cylindrical algebraic decomposition based on regular chains. *Journal of Symbolic Computation*, 75:74–93, 2016.

[19] A. Christov and D. Grigoriev. Complexity of quantifier elimination in the theory of algebraically closed fields. In *Proceedings of the MFCS 1984*, volume 176 of *LNCS*, pages 17–31. Springer, 1984.

[20] J.-F. Collard. *Reasoning About Program Transformations.* Springer, 2003.

[21] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20–23, 1975*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

[22] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, 1991.

[23] J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1–2):29–35, 1988.

[24] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

[25] A. Dolzmann. *Algorithmic Strategies for Applicable Real Quantifier Elimination.* Doctoral dissertation, Universität Passau, Germany, 2000.

[26] A. Dolzmann, O. Gloor, and T. Sturm. Approaches to parallel quantifier elimination. In *Proceedings of the ISSAC 1998*, pages 88–95. ACM, 1998.

[27] A. Dolzmann and T. Sturm. REDLOG: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, 1997.

[28] A. Dolzmann and T. Sturm. Simplification of quantifier-free formulae over ordered fields. *Journal of Symbolic Computation*, 24(2):209–231, 1997.

[29] A. Dolzmann and T. Sturm. Redlog user manual, 2nd edition. Technical Report MIP-9905, FMI, Universität Passau, Germany, 1999.

[30] A. Dolzmann, T. Sturm, and V. Weispfenning. A new approach for automatic theorem proving in real geometry. *Journal of Automated Reasoning*, 21(3):357–380, 1998.

[31] A. Dolzmann, T. Sturm, and V. Weispfenning. Real quantifier elimination in practice. In *Algorithmic Algebra and Number Theory*, pages 221–247. Springer, 1999.

[32] M. El Kahoui and A. Weber. Deciding Hopf bifurcations by quantifier elimination in a software component architecture. *Journal of Symbolic Computation*, 30(2):161–179, 2000.

[33] M. England, R. Bradford, and J. H. Davenport. Improving the use of equational constraints in cylindrical algebraic decomposition. In *Proceedings of the ISSAC 2015*, pages 165–172. ACM, 2015.

[34] H. Errami, M. Eiswirth, D. Grigoriev, W. M. Seiler, T. Sturm, and A. Weber. Efficient methods to compute Hopf bifurcations in chemical reaction networks using reaction coordinates. In *Proceedings of the CASC 2013*, volume 8136 of *LNCS*, pages 88–99. Springer, 2013.

[35] H. Errami, M. Eiswirth, D. Grigoriev, W. M. Seiler, T. Sturm, and A. Weber. Detection of Hopf bifurcations in chemical reaction networks using convex coordinates. *Journal of Computational Physics*, 291:279–302, 2015.

[36] H. Errami, T. Sturm, and A. Weber. Algorithmic aspects of Muldowney's extension of the Bendixson-Dulac criterion for polynomial vector fields. In *Polynomial Computer Algebra*, pages 25–28, St. Petersburg, Russia, 2011. The Euler International Mathematical Institute.

[37] K. Gatermann, M. Eiswirth, and A. Sensse. Toric ideals and graph theory to analyze Hopf bifurcations in mass action systems. *Journal of Symbolic Computation*, 40(6):1361–1382, 2005.

[38] D. Grigoriev. Complexity of deciding Tarski algebra. *Journal of Symbolic Computation*, 5(1–2):65–108, 1988.

[39] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38(1):173–198, 1931.

[40] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSAT: An efficient weighted Max-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.

[41] H. Hong. Comparison of several decision algorithms for the existential theory of the reals. Technical Report 91-41, RISC, Johannes Kepler University, Linz, Austria, 1991.

[42] H. Hong, R. Liska, and S. Steinberg. Testing stability by quantifier elimination. *Journal of Symbolic Computation*, 24(2):161–187, 1997.

[43] N. I. Ioakimidis. Automatic derivation of positivity conditions inside boundary elements with the help of the REDLOG computer logic package. *Engineering Analysis with Boundary Elements*, 23(10):847–856, 1999.

[44] K. Korovin, M. Košta, and T. Sturm. Towards conflict-driven learning for virtual substitution. In *Proceedings of the CASC 2014*, volume 8660 of *LNCS*, pages 256–270. Springer, 2014.

[45] K. Korovin, N. Tsiskaridze, and A. Voronkov. Conflict resolution. In *Proceedings of the CP 2009*, volume 5732 of *LNCS*, pages 509–523. Springer, 2009.

[46] M. Košta and T. Sturm. A generalized framework for virtual substitution. arXiv:1501.05826. 2015.

[47] M. Košta, T. Sturm, and A. Dolzmann. Better answers to real questions. *Journal of Symbolic Computation*, 74:255–275, 2016.

[48] A. Lasaruk. Parametrisches Integer-Solving. Master's thesis, Universität Passau, Germany, 2005.

[49] A. Lasaruk and T. Sturm. Weak quantifier elimination for the full linear theory of the integers. A uniform generalization of Presburger arithmetic. *Applicable Algebra in Engineering, Communication and Computing*, 18(6):545–574, 2007.

[50] D. Lazard. Quantifier elimination: Optimal solution for two classical examples. *Journal of Symbolic Computation*, 5(1–2):261–266, 1988.

[51] R. Loos and V. Weispfenning. Applying linear quantifier elimination. *THE Computer Journal*, 36(5):450–462, 1993.

[52] S. McCallum. *An Improved Projection Operation for Cylindrical Algebraic Decomposition.* Doctoral dissertation, University of Wisconsin-Madison, USA, 1984.

[53] S. McCallum. On projection in CAD-based quantifier elimination with equational constraint. In *Proceedings of the ISSAC 1999*, pages 145–149. ACM, 1999.

[54] S. McCallum. On propagation of equational constraints in CAD-based quantifier elimination. In *Proceedings of the ISSAC 2001*, pages 223–231. ACM, 2001.

[55] B. Mishra. *Algorithmic Algebra.* Texts and Monographs in Computer Science. Springer, 1993.

[56] G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.

[57] R. Niedermeier and P. Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. *Journal of Algorithms*, 47(2):63–77, 2003.

[58] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, 2006.

[59] M. Petkovšek, H. Wilf, and D. Zeilberger. $A = B$. A. K. Peters, 1996.

[60] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du Premier Congrès des Mathématiciens des Pays Slaves*, pages 92–101, Warsaw, Poland, 1929.

[61] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part III: Quantifier elimination. *Journal of Symbolic Computation*, 13(3):329–352, 1992.

[62] A. Seidl. Effiziente Realisierung reeller algebraischen Zahlen. Master's thesis, Universität Passau, Germany, 2001.

[63] A. M. Seidl and T. Sturm. Boolean quantification in a first-order context. In *Proceedings of the CASC 2003*, pages 329–345. Technische Universität München, Germany, 2003.

[64] A. Strzeboński. Computation with semialgebraic sets represented by cylindrical algebraic formulas. In *Proceedings of the ISSAC 2010*, pages 61–68. ACM, 2010.

[65] A. Strzeboński. Solving polynomial systems over semialgebraic sets represented by cylindrical algebraic formulas. In *Proceedings of the ISSAC 2012*, pages 335–342. ACM, 2012.

[66] A. W. Strzeboński. Cylindrical algebraic decomposition using validated numerics. *Journal of Symbolic Computation*, 41(9):1021–1038, 2006.

[67] T. Sturm. *Real Quantifier Elimination in Geometry.* Doctoral dissertation, Universität Passau, Germany, 1999.

[68] T. Sturm. Reasoning over networks by symbolic methods. *Applicable Algebra in Engineering, Communication and Computing*, 10(1):79–96, 1999.

[69] T. Sturm. An algebraic approach to offsetting and blending of solids. In *Proceedings of the CASC 2000*, pages 367–382. Springer, 2000.

[70] T. Sturm. Linear problems in valued fields. *Journal of Symbolic Computation*, 30(2):207–219, 2000.

[71] T. Sturm and A. Tiwari. Verification and synthesis using real quantifier elimination. In *Proceedings of the ISSAC 2011*, pages 329–336. ACM, 2011.

[72] T. Sturm and A. Weber. Investigating generic methods to solve Hopf bifurcation problems in algebraic biology. In *Proceedings of Algebraic Biology 2008*, volume 5147 of *LNCS*, pages 200–215. Springer, 2008.

[73] T. Sturm, A. Weber, E. O. Abdel-Rahman, and M. El Kahoui. Investigating algebraic and logical algorithms to solve Hopf bifurcation problems in algebraic biology. *Mathematics in Computer Science*, 2(3):493–515, 2009.

[74] T. Sturm and V. Weispfenning. Computational geometry problems in REDLOG. In *Proceedings of the ADG 1997*, volume 1360 of *LNCS*, pages 58–86. Springer, 1997.

[75] T. Sturm and V. Weispfenning. Rounding and blending of solids by a real elimination method. In *Proceedings of the IMACS 1997*, volume 2, pages 727–732. Wissenschaft & Technik Verlag, Berlin, 1997.

[76] T. Sturm and V. Weispfenning. Quantifier elimination in term algebras. The case of finite languages. In *Proceedings of the CASC 2002*, pages 285–300. Technische Universität München, Germany, 2002.

[77] T. Sturm and C. Zengler. Parametric quantified SAT solving. In *Proceedings of the ISSAC 2010*, pages 77–84. ACM, 2010.

[78] A. Tarski. A decision method for elementary algebra and geometry. Prepared for publication by J. C. C. McKinsey. RAND Report R109, August 1, 1948, Revised May 1951, Second Edition, RAND, Santa Monica, CA, 1957.

[79] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.

[80] A. Weber, T. Sturm, and E. O. Abdel-Rahman. Algorithmic global criteria for excluding oscillations. *Bulletin of Mathematical Biology*, 73(4):899–916, 2011.

[81] V. Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1–2):3–27, 1988.

[82] V. Weispfenning. Parametric linear and quadratic optimization by elimination. Technical Report MIP-9404, Universität Passau, Germany, 1994.

[83] V. Weispfenning. Quantifier elimination for real algebra—the cubic case. In *Proceedings of the ISSAC 1994*, pages 258–263. ACM, 1994.

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

[85] V. Weispfenning. Simulation and optimization by quantifier elimination. *Journal of Symbolic Computation*, 24(2):189–208, 1997.

[86] V. Weispfenning. Semilinear motion planning in REDLOG. *Applicable Algebra in Engineering, Communication and Computing*, 12(6):455–475, 2001.

[87] D. J. Wilson. *Advances in Cylindrical Algebraic Decomposition.* Doctoral dissertation, University of Bath, UK, 2014.