# Symbolic-Numeric Quantifier Elimination with Industrial Applications

岩根, 秀直

https://doi.org/10.15017/1441053

# Doctoral Thesis

## Symbolic–Numeric Quantifier Elimination with Industrial Applications

Guidance

Professor Hirokazu Anai

**Hidenao Iwane**

岩根 秀直

Graduate School of Mathematics

Kyushu University

January, 2014

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Acknowledgements

It would not have been possible to write this thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

First of all, I would especially like to thank my supervisor Professor Hirokazu Anai. He has offered many opportunities for me and his knowledge, kindness, support, patience, and vision have provided a good basis for this investigation.

I am deeply grateful to Professor Kazuhiro Yokoyama for his detailed comments and helpful advice.

I am deeply grateful to Professor Nobuki Takayama and Professor Masayuki Noro. They introduced me to computer algebra and have taught me computer programming skills.

I am also thankful to Dr. Hitoshi Yanami. His guidance helped me during my research and while writing papers.

I am thankful to Dr. Christoper W. Brown. He has given me helpful comments about my implementation of SyNRAC and taught me solution formula construction in quantifier elimination for cylindrical algebraic decomposition.

I am grateful to Professor Hoon Hong. He has been invaluable in improving my writing and presentation skills.

I thank Dr. Adam Strzeboński for suggesting the use of sign information in the lifting phase of cylindrical algebraic decomposition.

In addition, this research was supported in part by FUJITSU LABORATORIES LTD.

Finally, I give my special thanks to my family. Words cannot express how grateful I am to my parents for their love and lasting support. I am very much thankful to my wife, my daughter and my son for their love, understanding, and continued support while I have completed this thesis — they allowed me to spend most of the time on this thesis.

# Chapter 1

# Introduction

## 1.1 History and background

Many mathematical and engineering problems can be naturally translated to formulae consisting of polynomial equations, inequalities, quantifiers ($\forall$, $\exists$) and Boolean operators ($\wedge$, $\vee$, $\neg$, $\rightarrow$, etc). Such formulae construct sentences in the first-order theory of real closed fields (RCFs) and are called first-order formulae (FOFs). A *quantifier elimination* (QE) is an algorithm for computing an equivalent quantifier-free formula for a given FOF, and thus, QE is a very powerful concept for solving problems containing real algebraic constraints. For example, QE can exactly prove real theorems, perform geometric reasoning, network analysis, sizing and diagnosis, solve polynomial optimization problems, transportation problems, scheduling problems, mechanical engineering, and stability analysis, and so on.

In the 1930's, A. Tarski [79] showed that RCF allows QE: for any RCF formula $\phi(x_1, \ldots, x_r)$, there exists an equivalent quantifier-free formula $\psi(x_1, \ldots, x_r)$ in the same vocabulary. He also gave the first QE procedure for RCFs. For example, the formula $\exists x(x^2 + ax + b \leq 0)$ can be reduced to a quantifier-free formula $a^2 - 4b \geq 0$ by QE. If all variables are quantified, QE decides whether the given formula is true or false (this is a decision problem). However, his algorithm was totally impractical. Moreover, J.H. Davenport and J. Heints [24] proved that the asymptotic worst-case complexity of QE over the reals is doubly exponential with respect to the number of quantifier alternations in the input FOF. Therefore, to realize practical and effective methods using QE, the most significant issue is increasing the speed of the procedure.

In 1975, G.E. Collins [19] discovered a new method based on *cylindrical algebraic* *decomposition* (CAD) that was far more efficient than any previous approach. However, QE based on CAD is not considered to be practical on computers yet, since CAD usually consists of many purely symbolic computations and has high computational complexity. CAD, even restricted to a linear signature, is known to have a doubly exponential lower-bound [14], and problems containing only six variables may be hard for today's computers.

To circumvent the inherent computational complexity of a QE algorithm based on CAD, several researchers have focused on developing QE algorithms specialized to particular types of input formulae by exploiting this specificity; see [18, 82, 39, 80, 77].

V. Weispfenning and R. Loos [81, 55] gave algorithms for the cases where the bounded variables are only linear and quadratic. These algorithms have been developed and applied to various problems. In [42], H. Hong gave an algorithm for constructing a quantifier-free formula constrained by quadratic equations in the form $\exists x\ (a_2x^2 + a_1x + a_0 = 0 \wedge F(x, x_1, \ldots, x_n))$, where $a_2$, $a_1$ and $a_0$ are represented by $x_1, \ldots, x_n$ and the system $\{a_2 = a_1 = a_0 = 0\}$ has no solution.

For QE problems in the form $\forall x(f(x) > 0)$ where the degree of $f$ is an even integer, L. González-Vega [35] proposed a combinatorial algorithm based on the Sturm-Habicht sequence which is a theory on root classification of polynomials. The real root classification method [89] of a univariate polynomial with parameter coefficients can also apply the special QE problem. H. Anai and S. Hara [2] showed that many practical engineering problems such as control system design problems can be recast using a condition $\forall x(x \geq 0 \rightarrow f(x) > 0)$, which is called a sign definite condition (SDC), and proposed a special QE algorithm for SDCs.

Moreover, a one-block QE algorithm [40] has been proposed. Allowing measure-zero error in the output formula makes this algorithm efficient.

These directions are quite promising in practice since a number of important problems in engineering have been successfully reduced to such particular input formulae and resolved efficiently using the specialized QE algorithms. Examples of concrete successful applications are found in [83, 78, 29, 34, 2, 6, 87].

However, there still remain many significant problems in engineering that cannot be recast as such particular formulae. Therefore, it is highly desirable to develop an efficient algorithm for CAD, and many researchers have proposed improvements of the CAD algorithm, such as improved projection operators [19, 41, 58, 12], partially for CAD construction [21, 57, 74, 47], use of numerical methods [43, 73, 22, 66, 7, 75, 49], efficient projection order [27], use of equation constraints [59, 60, 74, 11], solution formula construction [44, 10], computing CAD via triangular decompositions [17, 16], preprocessing input formulae by Gröbner bases [85] and so on. Practically efficient software systems for QE have been developed on several computer algebra systems, such as QEPCAD [13], REDLOG [28], Mathematica [75] and SyNRAC [50].

Another promising direction is to use generalized critical values for global optimization, which originates from the scheme of sums of squares [72]. Within the framework of the critical point method (see [8] for details), which is proved to have computational complexity that is doubly exponential with respect to the number of alternates of quantifiers, generalized critical values were first used [69]. In the case of the sums of squares approach, we note that algebraic certificates are provided to compute lower bounds on the global infimum (see [64]). The computation of "symbolic" algebraic certificates (instead of "numerical" ones) has also been proposed [51]. Moreover, the problem of obtaining theorems on the existence of algebraic certificates based on sums of squares is caused by [37].

## 1.2 Our aim and approach

As described in the previous section, since many industrial applications can be naturally translated to FOFs, improvements of QE algorithms are required. In particular, we believe that improvements of CAD, which is a general QE algorithm, is one of the most important. This thesis focuses on further improvement of QE algorithms.

To achieve our goal, we use three approaches:

- First, we improve QE algorithms algebraically. In Section 2.3.2, we show avoidance of heavy symbolic computations by using previous computations. In Section 3.3, we show a necessary condition for a special QE algorithm and remove redundant formulae.

- Second, we improve QE algorithms by using numeric methods. We can avoid heavy symbolic computation by interval arithmetic techniques without loss of exactness, for example, real root isolation in an algebraic extension field. We show our numeric approaches in Sections 2.3.1, 2.4.1, 2.4.2 and 2.5.1.

- Finally, we focus on special types of input formulae for industrial applications.

We verify the effectiveness of our approaches by applying them to industrial problems.

## 1.3 Contents of this thesis

This thesis consists of five chapters, Chapters 1 through 5.

In Chapter 2, we consider an improvement of CAD, which is a general QE algorithm, by using symbolic–numeric methods without loss of exactness. We present an effective symbolic–numeric cylindrical algebraic decomposition (SNCAD) algorithm for QE incorporating several new devices, which we call *"quick tests"*. The simple quick tests are run beforehand to detect *unnecessary* procedures that may be skipped without violating the correctness of results, thus considerably reducing the computing time without loss of exactness. In this chapter, we utilize the structure of input formulae such that the truth value of a certain subset is easily decided by numerical computation. We see quite often but not always this structure in industrial problems. The effectiveness of the SNCAD algorithm is examined in a number of experiments including practical engineering problems, which also reveal the quality of the implementation. Experimental results show that our implementation significantly improves efficiency compared with our previous work.

In Chapter 3, we focus on one particular input formula,

$$\forall x \ (x \geq 0 \to f(x) > 0) \tag{1.1}$$

where $f(x)$ is a polynomial with real parameters, which we call a *sign definite condition (SDC)*. This formula is important because several practical engineering problems such as control system design problems can be recast as SDCs [3]. We note that we mainly consider the case where the coefficients of $f$ contain some parameters. An effective QE algorithm for SDCs was proposed in [45] based on a combinatorial approach using a real root counting technique. To improve the algorithm, simplification of the output formula is needed. For this purpose, we propose two approaches. First, we show a necessary condition for the SDC to simplify an output formula algebraically. The necessary condition enables us to eliminate extraneous sign combinations derived from real root counting using the Sturm-Habicht sequence. Second, we show an approach to simplify formulae using a logic minimization method. We obtain simple formulae by using the idea of *don't cares* for handling sign conditions that no real numbers satisfy. A *don't care* is an input where a function is not specified. Experimental results show that our approach significantly simplifies formulae.

In Chapter 4, we focus on optimization problems derived from some real engineering problems. When all constraints and objective functions are expressed as polynomials, the algebraic method QE based on a CAD algorithm can compute those as semi-algebraic sets. But, for aiming practical computation and dealing with non-polynomial objective functions, we propose a new method new optimization methods for classes based on a symbolic QE algorithm, combined with numerical computation, are proposed. The total efficiency of the design process is improved by reducing the number of numerical yield-rate evaluations. In addition, useful information such as the explicit relations among design variables, objective functions, and the yield rate, is provided.

Discussion and some concluding remarks are made in Chapter 5.

# Chapter 2

# Symbolic-Numeric Cylindrical Algebraic Decomposition

## 2.1 Introduction

Cylindrical algebraic decomposition (CAD) is a general-purpose symbolic method used in quantifier elimination (QE), and is an effective tool for solving real algebraic constraints (in particular, parametric and non-convex cases) arising in many engineering and industrial problems. However, QE based on CAD is not considered to be practical on computers, since CAD usually consists of many purely symbolic computations and has high computational complexity.

A CAD algorithm consists of three phases, namely the projection phase, the base phase and the lifting phase, and uses symbolic and algebraic computations to obtain exact results.

The main improvements of CAD have been achieved in the projection phase by focusing on the projection operator (e.g., [19, 41, 58, 12]), because this is the most effective way to reduce the computational time. Computational difficulties in the lifting phase stem from symbolic computation over towers of algebraic extensions and combinatorial explosion in CAD construction.

An effective method of CAD construction is to use numerical computation, rather than symbolic treatment, with as much derived numerical information for algebraic numbers as possible without violating the correctness of the results. Existing attempts to introduce numerical computation into CAD construction include [43, 73, 22, 66, 7, 75, 49]. In general, to achieve more effective and practical implementation of algebraic algorithms, the research has pursued hybrid methods combining computer algebra with numerical verification [67, 68]. From the viewpoint of symbolic–numeric CAD methods, one can see that such an approach provides new validated numerical methods for non-convex optimization problems, with the help of symbolic computation, that are difficult to treat using ordinary numerical methods. Both aspects are surely of practical importance.

Additionally many improvements of the CAD algorithm tailored to QE problems have been proposed. Partial CAD [21] avoids unnecessary lifting procedures by evaluating truth values and using quantifier information of input formulae. This reduces the computing time in the lifting phase. Furthermore, many approaches for special types of input formulae have been proposed, such as those found in [57, 59, 60, 74, 11].

Meanwhile, turning to optimization problems, there have been several attempts to develop symbolic–numeric algorithms for solving polynomial optimization problems (POPs). Efficient algorithms based on CAD have been developed to exactly solve semidefinite programming [5] and to construct explicit optimal value functions [47].

Although QE based on CAD may at a glance seem far from practical applications owing to its inefficient computation, with doubly exponential behavior in the worst case, extensive research has been done to improve its efficiency as mentioned above and some progress has been made. There is still plenty of room for further exploration of possible refinement in view of practical efficiency and implementation methodology. This chapter focuses on further improving the practical efficiency of CAD and extending its practical applicability to science and engineering problems. Our main focus is on two time-consuming phases, namely the projection phase and the lifting phase.

We thus employ as our basis a scheme for symbolic–numeric cylindrical algebraic decomposition (SNCAD) that was roughly introduced (without detailed procedures) in [7]. In the lifting phase, the scheme uses certified numerical computation over algebraic extensions and the dynamic evaluation technique in [26] with successive representations of algebraic extensions. For efficient realization of the scheme, we refine it by incorporating several new devices, which we call *"quick tests"* (see Section 2.2), in a concrete way shown in the following sections. We employ quick tests because we expect that one can often detect an *unnecessary* procedure that may be skipped without violating the correctness of the results by running a simple test beforehand.

Here, we outline the results obtained in this chapter. The main computational difficulties in the lifting phase are symbolic computation over algebraic extension fields and combinatorial explosion in making stacks. We devise quick tests to avoid symbolic computation over algebraic extension fields using numerical computation (see Section 2.3) and to avoid making stacks irrelevant to the output before actually making a stack (see Section 2.4). We confirm that these tests save computing time in many practical examples (see Section 2.7). In the projection phase, the explosion of projection factors is a crucial issue. We propose a test to verify if a polynomial in a projection factor can be removed by exploiting the structure of the input formula (see Section 2.5). It turns out that this approach works well for many practical engineering problems, in particular for POPs (see Section 2.7). All proposed improvements are implemented in the Maple package SyNRAC [50] to solve real algebraic constraints.

The rest of this chapter is organized as follows. Section 2.2 provides an outline of our strategy for improving the practical efficiency of QE by CAD. The scheme for a standard CAD algorithm and QE based on CAD are also briefly reviewed in Section 2.2. Section 2.3 presents quick tests using numerical computation that avoid symbolic computation over an algebraic extension field in the lifting phase of CAD. Tests for reducing the number of stacks in the lifting phase are described in Section 2.4. Section 2.5 explains a technique for decreasing the number of projection factors by exploiting the structure of an input formula. Section 2.6 shows procedures for our CAD algorithm. Behavior analysis of the proposed improvements for speeding up QE is discussed using the experimental results reported in Section 2.7. Concluding remarks are made in Section 2.8.

## 2.2 Our Strategy for Speeding up CAD

In this section we introduce our strategy for improving the efficiency of QE based on CAD. We begin by showing a brief sketch of Collins's CAD algorithm, which plays a central role in his QE. We then state our strategy which consists of six quick tests. Each quick test is thoroughly described later in Sections 2.3 through 2.5. We give a full description of the Collins CAD algorithm at the end of the section to clarify the terms used from the next section on.

The Collins QE algorithm based on CAD [19] is a general-purpose QE first published in 1975. Since then, there have been many papers that have improved the algorithm or that have discussed its computational complexity. Although it has been proved that CAD-based QE has in the worst case doubly exponential behavior [24, 14], there is much room for enhancing its efficiencies from practical and implementational points of view. We propose a strategy for improving the Collins QE algorithm, part of which has already been presented [7]. We confirm the effectiveness of our implementation by investigating statistical data for many example problems in Section 2.7. We stress that our algorithm works effectively for many practical problems coming in engineering and scientific fields.

### 2.2.1 Outline of CAD and its Computational Problems

We briefly outline the Collins CAD algorithm, the most important subalgorithm of the Collins QE. Let us denote the fields of rational numbers and real numbers by $\mathbb{Q}$ and $\mathbb{R}$, respectively. We assume that we are given as an input formula for QE a prenex first-order formula $\varphi$ over the elementary theory of real closed fields:

$$\varphi(x_1, \ldots, x_r) \equiv \mathsf{Q}_{q+1}x_{q+1} \cdots \mathsf{Q}_r x_r \ \psi(x_1, \ldots, x_r),$$

where $\mathsf{Q}_j \in \{\exists, \forall\}$ and $\psi$ is a quantifier-free formula. Note that $x_1, \ldots, x_q$ are called *free variables* and $x_{q+1}, \ldots, x_r$ *quantified variables.* A QE algorithm takes $\varphi$ as an input and returns a quantifier-free formula in the free variables that is equivalent to the input. We can assume, without loss of generality, that every atomic formula in $\psi$ is represented in the form $f \, \rho \, 0$, where $f \in \mathbb{Q}[x_1, \ldots, x_r]$ is a polynomial with rational coefficients on $x_1, \ldots, x_r$ and $\rho$ is a relational operator in $\{\leq, <, =, \neq\}$. Winkler [86] calls such an expression $\varphi$ a *standard* formula.

A CAD algorithm takes a set of $r$-variate polynomials $F_r$ as an input and returns a partition of the $r$-dimensional real space $\mathbb{R}^r$ into sign-invariant subsets with respect to $F_r$. For the standard input formula $\varphi$ above the CAD input consists of the left-hand-side polynomials collected from its free part $\psi$. A CAD algorithm itself consists of three phases: projection, base and lifting phases.

A projection procedure takes a finite subset of $\mathbb{Q}[x_1, \ldots, x_r]$ and computes a family of subsets $\{F_i\}_{i=1,\ldots,r-1}$, where $F_i \subset \mathbb{Q}[x_1, \ldots, x_i]$. Any factor of a polynomial in $F_i$ is called a *projection factor.* A projection operator applied to the CAD input $F_r \subset \mathbb{Q}[x_1, \ldots, x_r]$ returns a set $F_{r-1} \subset \mathbb{Q}[x_1, \ldots, x_{r-1}]$ and it is repeatedly applied to the preceding output till a set $F_1 \subset \mathbb{Q}[x_1]$ of univariate polynomials is obtained. Note that a projection operator should be defined so that $\{F_i\}_{i=1,\ldots,r}$ has some property, which is briefly explained in Section 2.2.2.

In the base phase a base procedure carries out real root isolation for $F_1$. The real line $\mathbb{R}^1$ is partitioned into the set of the real roots in $F_1$ and the remaining open intervals.

Lifting proceeds in the opposite direction of projection, constructing a partition of $\mathbb{R}^{i+1}$ from that of $\mathbb{R}^i$. Let $C$ be a member, called a *cell*, in a partition of $\mathbb{R}^i$. A cylinder $C \times \mathbb{R}$ can be partitioned into sign-invariant subsets for $F_{i+1}$, called a stack on $C$. This is realized in a cylindrical manner: i.e., the space is cut by the graphs of $F_{i+1}$, each component of which is homeomorphic to $C$. This is always the case owing to the requirement of the projection operator. The components as well as the remaining cylinders, piled up alternately, consist of a partition of $C \times \mathbb{R}$. See Section 2.2.5 in more detail. A union of the stacks on all the cells in $\mathbb{R}^i$ becomes a partition of $\mathbb{R}^{i+1}$ and this process is repeated until we obtain a partition of the $r$-dimensional space $\mathbb{R}^r$, the output of CAD. Figure 2.1 illustrates the three phases of CAD.

In an actual lifting procedure a sample point $s \in C$ is chosen and substituted for $(x_1, \ldots, x_i)$ in each of the polynomials $F_{i+1}$ to obtain a set of univariate polynomials on $x_{i+1}$. Root isolation is applied for the set as in the base phase and all the real roots found determine the structure of a stack over $C$.

$$\begin{array}{lcl}
F_r \subset \mathbb{Q}[x_1, \ldots, x_r] & \longrightarrow & \mathbb{R}^r = \bigsqcup C^{(r)} \\
\downarrow \text{projection} & & \uparrow \text{lifting} \\
F_{r-1} \subset \mathbb{Q}[x_1, \ldots, x_{r-1}] & & \mathbb{R}^{r-1} = \bigsqcup C^{(r-1)} \\
\downarrow \text{projection} & & \uparrow \text{lifting} \\
\vdots & & \vdots \\
\downarrow \text{projection} & & \uparrow \text{lifting} \\
F_2 \subset \mathbb{Q}[x_1, x_2] & & \mathbb{R}^2 = \bigsqcup C^{(2)} \\
\downarrow \text{projection} & \text{base} & \uparrow \text{lifting} \\
F_1 \subset \mathbb{Q}[x_1] & \longrightarrow & \mathbb{R}^1 = \bigsqcup C^{(1)}
\end{array}$$

Figure 2.1: Flow of CAD

Once a partition of the full space $\mathbb{R}^r$ into sign-invariant subsets with respect to $F_r$ is obtained, it is straightforward to construct the output for QE by evaluating a sample point from each cell; a Boolean expression is constructed by collecting the quantifier-free formulae for appropriate regions. This is called the solution formula construction phase, and it is the last part of the QE algorithm.

Of the three phases in CAD, the most time-consuming part is the lifting phase. We ascribe the difficulty in the lifting phase to three sources: (i) symbolic computation over algebraic extension fields, (ii) combinatorial explosion in making stacks and (iii) superfluous projection factors.

To avoid symbolic computation in an extension field we use numerical computation as far as its result is confirmed. Simple, quick numerical computation can tell us if precise symbolic computation is needed and in many cases numerical computation suffices. This preceding trial saves us from unnecessary symbolic computation.

To reduce cell lifting, we deploy two types of procedures. One is used in the lifting phase before the lifting of a cell. The other is used in the projection phase to reduce projection factors. Reducing the number of stacks in lifting saves computing time. A full decomposition of $\mathbb{R}^r$ is not always needed to compute a quantifier-free equivalent of the input. We carry out tests before making a stack, which allows us to avoid making stacks that are irrelevant to the output. A smaller number of projection factors results in a smaller number of real roots in real root isolation in the base and lifting phases, which also reduces computing time. Information on the input formula might be exploited to detect needless projection factors. We thus prepare tests to check if a polynomial in a projection factor can be removed.

On the above basis, we believe that it is worth carrying out a simple test that takes little time before carrying out a procedure that might be skipped without affecting the exactness of the output and that would probably take considerable time if performed, or that detects superfluous projection factors in advance. We prepare six such *quick tests* in total. Each quick test is outlined in the following three subsections. Detailed descriptions are found in Sections 2.3 through 2.5. From a practical viewpoint, these tests work satisfactorily in most example problems that we have tried. The effectiveness of the quick tests is fully investigated in Section 2.7.

### 2.2.2 Quick Tests for Reducing Symbolic Computation

In the lifting phase, every algebraic number appears as a root of a univariate polynomial over an algebraic number field. That algebraic number field itself has been generated by the algebraic numbers adjoined by real root isolation in previous liftings. A typical way of dealing with computation on an extension field is to use a primitive element of the field over $\mathbb{Q}$, represented by its minimal polynomial with an isolating interval. However, computing a primitive element in towers of extensions is very heavy, because factorization and GCD computation over algebraic number fields are required. We adopt a successive representation for towers of extension fields with the dynamic evaluation technique in [26]. We use a 'defining' polynomial, which is, unlike the minimal polynomial, not necessarily irreducible, with an isolating interval to identify an algebraic number.

We use numerical quick tests for a possible short cut to determining the sign of an algebraic expression. Isolating intervals are used to approximately evaluate an algebraic number using interval arithmetic techniques. In the lifting phase, instead of a univariate polynomial over an algebraic extension field, we deal with a univariate polynomial whose coefficients are intervals. We call such a polynomial an *interval polynomial.* To carry out real root isolation for interval polynomials, we need to decide whether a polynomial expression of algebraic numbers is zero. This procedure is heavy if implemented purely symbolically. Instead we substitute intervals for respective algebraic numbers to obtain an interval in which the exact value lies. It is possible to determine the sign of a number if the number is actually nonzero and the intervals have sufficiently high precision. We call this quick test *sign determination using interval arithmetic* (Section 2.3.1).

We remark that, in our computation, it is necessary for a defining polynomial to be square free. We prepare a *quick test for square freeness for interval polynomials* (Section 2.3.1). We carry out symbolic computation only if this quick test fails.

The sign information for a cell in $\mathbb{R}^i$ for a projection factor can be reused when carrying out real root isolation of an interval polynomial at the above level. These data are stored and looked up later in a *quick test using a previous computation* (Section 2.3.2).

### 2.2.3 Quick Tests for Reducing Stack Construction

Constructing a stack over a cell $C$ in $\mathbb{R}^i$ partitions the cylinder $C \times \mathbb{R}^i$. The number of cells dramatically grows as the lifting phase proceeds and the nested structure easily results in combinatorial explosion and prevents a CAD procedure from completing a partition of the full space $\mathbb{R}^r$.

To complete full CAD computation, every cell in $\mathbb{R}^i$, $i < r$, should be lifted to a set of cells in $\mathbb{R}^{i+1}$ until we obtain a partition of the full space $\mathbb{R}^r$. However, it might be unnecessary for a QE algorithm to construct a full CAD. What we need to do is not decompose $\mathbb{R}^r$ but compute a quantifier-free equivalent of the QE input. To improve efficiency, it is important not to lift the cells that are irrelevant to the QE output. We can use cell information to check if further computation is needed. From this point of view, we can skip a considerable number of cell liftings and still obtain a correct output. G.E. Collins and H. Hong proposed partial CAD [21], based on the above idea. Before lifting a cell, they try evaluating an atomic formula on it. If the evaluation turns out to be a truth value, the information is used to judge if the cell is lifted. They called this evaluation before lifting a *trial evaluation.* To improve efficiency, this test should be done before merging and sorting the real roots found (Section 2.4.1).

We propose quick tests using the information on the variables in the input formula. When, for example, the input formula has a quantified variable that is bounded by an interval, it is not at all necessary to make stacks outside the interval. This can be checked by a quick test called *trial evaluation for bounded CAD* (Section 2.4.2).

If the input formula is existential and it implies some equational constraint $f = 0$, the polynomial $f$ is used to check if a cell lifting can be skipped. We call such a polynomial $f$ a *section polynomial* and prepare a *quick test using section polynomials* (Section 2.4.2). We note that the quick tests in this section are symbolic–numeric computations because they call quick test subroutines described in in Section 2.2.2.

### 2.2.4 Quick Tests for Reducing the Number of Projection Factors

A projection operator takes a subset of $i$-variate polynomials as input and returns a subset of $(i-1)$-variate polynomials. The lifting phase produces cells corresponding to the distinct real roots and the regions between those roots. This implies that reducing the number of projection factors helps reduce the number of cell liftings: i.e., it is important to remove irrelevant polynomials from the projection factors in as an early stage as possible. A constraint in the input represented by an interval might be used to reduce the number of projection factors. We prepare a *quick test for removing projection factors using interval constraints* (Section 2.5.1).

### 2.2.5 Cylindrical Algebraic Decomposition

We describe cylindrical algebraic decomposition in more detail to prepare the terms used in the rest of this chapter; see [19] for a full description. Assume that we are given a prenex standard formula $\varphi$ as in Section 2.2.1:

$$\varphi(x_1, \ldots, x_r) \equiv \mathsf{Q}_{q+1}x_{q+1} \cdots \mathsf{Q}_r x_r \; \psi(x_1, \ldots, x_r),$$

where $\mathsf{Q}_j \in \{\exists, \forall\}$ and $\psi$ is a quantifier-free formula. Let $F \subset \mathbb{Q}[x_1, \ldots, x_r]$ be the set of polynomials appearing in $\psi$ as the left hand sides of atomic formulae. A subset

$C \subseteq \mathbb{R}^r$ is said to be *sign-invariant* for $F$ if every polynomial in $F$ has a constant sign on all the points in $C$; i.e., $\psi(s)$ is either "true" or "false" for all $s \in C$.

Suppose we have a finite sequence $\mathcal{D}_1, \ldots, \mathcal{D}_r$ for $F$ that has the following properties.

1. Each $\mathcal{D}_i$ is a partition of $\mathbb{R}^i$ into finitely many connected semi-algebraic sets called *cells.*

2. $\mathcal{D}_{i-1}$, $1 < i \leq r$, consists exactly of the projections of all cells in $\mathcal{D}_i$ along the coordinate of the $i$-th variable in $(x_1, \ldots, x_r)$. For each cell $C \in \mathcal{D}_{i-1}$, we can determine its preimage $P(C) \subseteq \mathcal{D}_i$ under the projection.

3. Each cell $C \in \mathcal{D}_r$ is sign-invariant for $F$. Moreover for each cell $C \in \mathcal{D}_r$, we are given a *sample point* $s \in C$ in such a form that we can determine the sign of $f(s)$ for each $f \in F$ and thus evaluate $\varphi(s)$.

The partition $\mathcal{D}_r$ of $\mathbb{R}^r$ for $F$ is then called an *$F$-invariant cylindrical algebraic decomposition* of $\mathbb{R}^r$. A CAD algorithm computes such a sequence $\mathcal{D}_1, \ldots, \mathcal{D}_r$ and it consists of three phases: the *projection phase*, *base phase*, and *lifting phase.*

**Projection Phase:** We first construct from $F \subset \mathbb{Q}[x_1, \ldots, x_r]$ a new finite set $F' \subset \mathbb{Q}[x_1, \ldots, x_{r-1}]$ that satisfies a special condition called *delineability*, where the order of the real roots of all polynomials in $F$ as univariate polynomials in $x_r$ does not change above each connected subsets of $\mathbb{R}^{r-1}$ on which polynomials of $F'$ have constant signs.

The step constructing $F'$ from $F$ is called a *projection* and is denoted by $F' := \textsc{Proj}(F, x_r)$. We call polynomials in $F'$ *projection polynomials* and their irreducible factors *projection factors.* Iterative application of the operator $\textsc{Proj}$ leads to a finite sequence

$$F_r, \ldots, F_1, \quad \text{where} \ \ F_r := F, \ F_i := \textsc{Proj}(F_{i+1}, x_{i+1})$$

for $1 \leq i < r$. The operator $\textsc{Proj}$, in general, computes certain coefficients, discriminants, and resultants derived from polynomials in $F_{i+1}$ and their higher derivatives by regarding those as univariate polynomials in $x_{i+1}$. The final set $F_1$ consists of univariate polynomials in $x_1$.

**Base Phase:** In the base phase, we construct a partition $\mathcal{D}_1$ of one-dimensional real space $\mathbb{R}^1$ into finitely many intervals that are sign-invariant for $F_1$. This step is implemented by isolating real roots of the univariate polynomials in $F_1$. The real roots of the polynomials in $F_1$ are symbolically computed. That means each root $s$ is expressed as a pair of a polynomial $f \in \mathbb{Q}[x_1]$ and an interval $I$, where $f$ has a unique real root in $I$. We call such an interval an *isolating interval* of $s$.

**Lifting Phase:** The partitions $\mathcal{D}_{i+1}$ of $\mathbb{R}^{i+1}$ for $1 \leq i < r$ are computed recursively. The roots of all polynomials in $F_{i+1}$ as univariate polynomials in $x_{i+1}$ are delineated above each connected cell in a CAD produced from $F'$. Thus, we can cut the *cylinder* above $C$ into finitely many connected semi-algebraic sets (cells). This is done in the following way. Take a sample point $s = (s_1, \ldots, s_i) \in C$ and substitute $s$ for $(x_1, \ldots, x_i)$ in every polynomial in $F_{i+1}$, and obtain a set $L_{i+1}$ of univariate polynomials in $x_{i+1}$. Apply the real root isolation for the polynomials in $L_{i+1}$ and merge and sort the results.

A finite sequence $\mathcal{D}_1, \ldots, \mathcal{D}_r$ for $F$ has a tree structure: The first level of *nodes* under the root of the tree corresponds to the cells in $\mathcal{D}_1$. The second level of nodes stands for the cells in $\mathcal{D}_2$; i.e., the cylinders over the cells of $\mathbb{R}^1$. The *leaves* represent the cells of $\mathcal{D}_r$; i.e., a CAD of $\mathbb{R}^r$. A sample point of each cell is stored in its corresponding node or leaf.

## 2.3 Avoidance of Symbolic Computations

In CAD construction, many symbolic and algebraic computations are required and they tend to be very time consuming. Thus avoidance of symbolic computation can make CAD construction efficient. In this section, we present quick tests carried out to avoid symbolic computation using numeric computation and sign information obtained in CAD construction. We provide some definitions and notations here.

**Definition 1.** *Let $f$ be a non-constant polynomial in $\mathbb{Q}[x_1, \ldots, x_r]$. The* level *of $f$ is defined as the largest integer $k$ such that the degree in $x_k$ of $f$ is positive. That is, for a polynomial $f$ of the level $k$, $f \in \mathbb{Q}[x_1, \ldots, x_{k-1}, x_k]$ but $f \notin \mathbb{Q}[x_1, \ldots, x_{k-1}]$.*

Let $f$ be a polynomial in $\mathbb{Q}[x_1, \ldots, x_k, x_{k+1}]$, $c$ a polynomial in $\mathbb{Q}[x_1, \ldots, x_k]$, and $s = (s_1, \ldots, s_k)$ a point in $\mathbb{R}^k$ which is a $k$-tuple of algebraic numbers. We denote $f(s_1, \ldots, s_k, x_{k+1})$ and $c(s_1, \ldots, s_k)$ by $f(s, x_{k+1})$ and $c(s)$, respectively.

For $k \geq 1$ an algebraic number $s_k$ is also expressed by a pair of a polynomial $f_k \in \mathbb{Q}[x_1, \ldots, x_k]$ and an isolating interval $I_k$, where $s_k$ is a real root of $f_k(s_1, \ldots, s_{k-1}, x)$ uniquely in $I_k$. From now on, we identify $s_k \in \mathbb{R}$ with a pair $(f_k, I_k)$ and simply write $s_k$ for $(f_k, I_k)$.

**Definition 2.** *Let $I = [a, b]$ be a closed interval in $\mathbb{R}$. The* precision *of $I$ is defined as $b - a$.*

**Definition 3.** *For a real root $r$ of a polynomial, when a closed interval $I$ that contains $r$ is given, the* precision *of $I$ is set as that of $r$.*

The precision of $r$ is said to be improved when we choose another isolating interval $I'$ for $r$ whose precision is finer than that of $I$. For a closed, bounded interval $I$, the lower and upper boundaries are denoted $I^{(l)}$ and $I^{(u)}$, respectively.

### 2.3.1 Avoidance of Unnecessary Symbolic Computations by Numeric Computation

Here we present quick tests to avoid unnecessary symbolic computation using numeric computation.

#### Real Root Isolation and Sign Determination over Algebraic Extension Fields

If we use only symbolic computation in the lifting phase, the calculations must be performed in an algebraic extension field, which is computationally expensive.

In each step of CAD we have to find the real roots of a polynomial $f(x_1, \ldots, x_k, x_{k+1})$ at $s = (s_1, \ldots, s_k)$, where each $s_i$ is an algebraic number, given as a real root of another polynomial, or to determine the sign of such a polynomial $f$ at $s$. To do the above symbolically, we need heavy algebraic computation such as a tower of algebraic extension fields. However, we can replace such heavy computation with numerical computation using interval arithmetic techniques.

We introduce an interval polynomial as follows: for a polynomial $f(x_1, \ldots, x_k, x_{k+1})$ with degree $d$ algebraic numbers $s_i$ $(1 \leq i \leq k)$ with its isolating interval $I_i$ we consider a univariate polynomial $f(I_1, \ldots, I_k, x_{k+1}) = \sum_{i=1}^{d} [l_i, u_i] x_{k+1}^i$ obtained by substituting $I_i$ for $x_i$. The polynomial $f(I_1, \ldots, I_k, x_{k+1})$ is "rough evaluation" of $f$ at $s$ used to find real roots of $f(s, x_{k+1})$. We note that an interval polynomial $\tilde{f}(x_{k+1}) = f(I_1, \ldots, I_k, x_{k+1})$ can be thought of as a set of polynomials:

$$\tilde{f}(x_{k+1}) = \left\{ \sum_{i=0}^{d} c_i x_{k+1} \in \mathbb{R}[x_{k+1}] \,|\, c_i \in [l_i, u_i] \text{ for } i = 0, \ldots, d \right\}.$$

Using this definition, a usual univariate polynomial can be considered an interval polynomial according to $c_i = [c_i, c_i]$.

We consider a set $L$ of closed intervals, where each $J \in L$ contains only one root in $f(s, x_{k+1})$ without multiplicities counted and $L$ is pairwise disjoint. We call each $J$ of $L$ an *isolating interval* of a real root of $f$.

To obtain isolating intervals of real roots for an interval polynomial, it is required that the leading coefficient interval does not contain zero. In our setting, we consider the real root isolation in lifting a cell $C^{(k)} \in \mathcal{D}_k$ as follows. The degree of $f \in F_{k+1}$ is invariant on all points in $C^{(k)}$ because of the delineability. To determine the degree of $f(x_1, \ldots, x_k, x_{k+1}) = \sum_{i=0}^{d} c_i(x_1, \ldots, x_k)x_{k+1}^i$ at a sample point $s \in \mathbb{R}^k$ for $C^{(k)}$, we determine the sign of $c_i(s)$ from $i = d$ by decreasing order while $c_i(s)$ is equal to zero.

We can determine the sign of a polynomial $c(x_1, \ldots, x_k)$ at $s \in \mathbb{R}^k$ using Algorithm 1. To do so, we make good use of numerical computation, by which we can avoid symbolic zero determination.

```
Algorithm 1 SubstSamplePointCoef($c$, $s$)
Input: polynomial $c$ in $\mathbb{Q}[x_1, \ldots, x_k]$, sample point $s = (s_1, \ldots, s_k)$ in $\mathbb{R}^k$
Output: interval $J$, which substitutes $s_j$ for $x_j$ in $c$ where $J^{(l)} \cdot J^{(u)} > 0$ or $J^{(u)} =
    J^{(l)} = 0$ ($J \not\ni 0$ or $J = [0, 0]$)
  $J \leftarrow$ substitute each isolating interval of $s_j$ for $x_j$ in $c$
  ($J \leftarrow c(I_1, \ldots, I_k)$)
  if $J^{(l)} \leq 0$ and $J^{(u)} \geq 0$ then
    if SymbolicZeroChk($c, s$) then

      return $[0, 0]$
    end if
  end if
  while $J^{(l)} \leq 0$ and $J^{(u)} \geq 0$ do
    $s \leftarrow$ IncreasePrecision($s$)
    $J \leftarrow c(I_1, \ldots, I_k)$
  end while

  return $J$
```

In this algorithm, we first substitute the isolating intervals $I_i$ for $x_i$ in $c$ using interval arithmetic, and we then obtain the closed interval $J$, which contains $c(s)$. If $J$ does not contain zero, we can determine the sign of $c(s)$ correctly through only numeric computation. When $J$ contains zero, we switch to the symbolic computation SymbolicZeroChk($c$,$s$) to check whether $c$ is exactly zero at $s$. If $c$ is found not to equal zero at $s$, we only improve the precision of $s$ while $J$ contains zero to determine the sign of $c(s)$.

**Remark 4.** *In our implementation we use a recursive expression for a multivariate polynomial. When we evaluate the range of a polynomial by applying the interval arithmetic, we apply the Horner's scheme.*

**Remark 5.** *Since extension of intervals is generally restrained, each coefficient $I = [I^{(l)}, I^{(u)}]$ of an input interval polynomial is sign definite (i.e., $I^{(l)} \cdot I^{(u)} > 0$ or $I^{(l)} = I^{(u)} = 0$) in our implementation.*

### Avoidance of Unnecessary Factorization

If an algebraic number is not equal to zero, we can decide the sign of the algebraic number using only numeric computation. However, as shown in Algorithm 1, we need symbolic computation for zero determination because numeric computation cannot always determine exactly. As algebraic numbers are recursively given, we need field extension computation. To carry out such computation precisely, we need heavy symbolic computations such as factorization over an extension field or prime decomposition of an ideal.

We use dynamic evaluation (DE) [26, 30] to represent algebraic extensions in our scheme, by which we can avoid heavy computation of the algebraic factorization of defining polynomials, because DE requires only the square freeness of the defining polynomials. Square-freeness tests are known to be much easier to carry out than irreducibility tests.

**Proposition 6.** *We use the same notation as in the previous subsubsection. If $f$ is square free at $s$, each polynomial in $\tilde{f}(x_{k+1})$ is also square free, and the coefficient intervals of $\tilde{f}(x_{k+1})$ have enough precision, then the isolating intervals of real roots of $f$ can be computed by interval arithmetic.*

In fact, we can employ a concrete method IntvRealRootIsolSqfr based on Krawczyk's method [52]; see [49] for details.

**Remark 7.** *Under the assumption, $f$ is delineable on $I_1 \times \cdots \times I_k$. Therefore, the number of the real roots is constant.*

To avoid square-free decomposition in an extension field, we use a numeric real root isolation algorithm. The method IntvRealRootIsolSqfr outputs isolating intervals $J_1, \ldots, J_t$ of the real roots of an interval polynomial $\tilde{f}(x)$, which are pairwise disjoint, and for all $f(x) \in \tilde{f}(x)$, $f$ has a unique real root in $J_i$. When we apply the method IntvRealRootIsolSqfr to an interval polynomial $\tilde{f}(x_{k+1})$ obtained by substituting each isolating interval $I_i$ of $s_i$ for $x_i$ in $f(x_1, \dots, x_k, x_{k+1})$ computed by interval arithmetic, the method IntvRealRootIsolSqfr returns an error if

1. the polynomial $f(s, x_{k+1})$ is not square free, or
2. the coefficient intervals of the interval polynomial $\tilde{f}(x_{k+1})$ do not have enough precision.

Therefore, we have to call a symbolic square free decomposition method SymbolicSqfr for $f(s, x_{k+1})$ only when IntvRealRootIsolSqfr($\tilde{f}(x_{k+1})$) fails.

Algorithm 2 isolates real roots of a polynomial $f(s, x_{k+1})$, and IncreasePrecision increases the precision of the isolating interval of each coordinate of a sample point $s$.

**Algorithm 2** IntvRealRootIsol($f$, $s$)

```
Input: polynomial $c$ in $\mathbb{Q}[x_1, \dots, x_k]$, sample point $s = (s_1, \dots, s_k)$ in $\mathbb{R}^k$
Output: isolating intervals of the real roots of $f(s, x_{k+1})$
  $\tilde{f} \leftarrow$ substitute each isolating interval of $s_j$ for $x_j$ in $f$
  $L_f \leftarrow$ IntvRealRootIsolSqfr($\tilde{f}$)
  if error then
    $f \leftarrow$ SymbolicSqfr($f$)
    loop
      $\tilde{f} \leftarrow$ substitute each isolating interval of $s_j$ for $x_j$ in $f$
      $L_f \leftarrow$ IntvRealRootIsolSqfr($\tilde{f}$)
      if not error then
        break
      end if
      $s \leftarrow$ IncreasePrecision($s$)
    end loop
  end if

  return $L_f$
```

**Remark 8.** *Since we only deal with the real roots of $f(s, x_{k+1})$ in CAD construction, DE works correctly when $f(s, x_{k+1})$ is not square free but has only simple real roots. In this case* SymbolicSqfr *is not used in this algorithm. Hence the method* IntvRealRootIsol *guarantees square freeness of only the real roots of $f(s, x_{k+1})$.*

**Example 9.** *(adam1 [75])*

$$\forall x \forall y \ ((x < 0 \wedge x^2 + y^2 < \frac{99438}{100000}) \Rightarrow f(x, y) < 0),$$

*where*

$$R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \frac{z^5}{120} + \frac{z^6}{600},$$
$$f(x, y) = R(x + jy)R(x - jy) - 1,$$

*where $j$ denotes the imaginary unit.*

*In the base phase, by isolating the real roots of projection factors $F_1$, we obtain the sample points as shown in Figure 2.2. Filled circles and open circles show the section cells and sector cells, respectively.*

*Let $g(x)$ be the discriminant of $f(x, y)$ with respect to $y$. We obtain sample points $x_{(2)} = (g, [-3.33, -3.32])$, $x_{(4)} = (g, [-3.31, -3.30])$, and $x_{(6)} = (g, [-1.88, -1.87])$ from the condition that $f(x, y)$ has a multiple root. However, $f(x, y)$ has a multiple real root only when $x = x_{(4)}$. In particular, since $f(x_{(2)}, y)$ does not have a real root, we can avoid factorization over an algebraic extension field even if we do not use dynamic evaluation.*

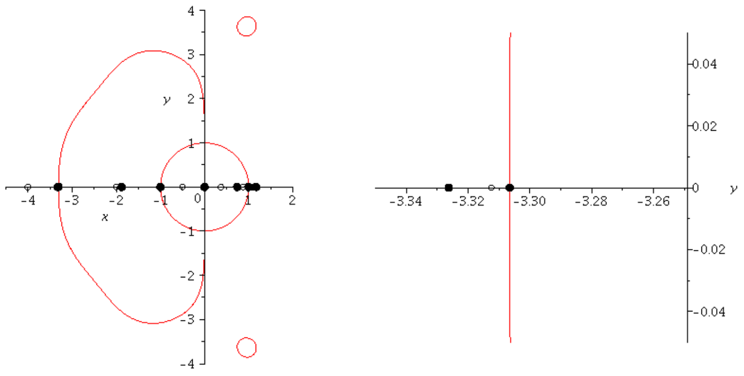

Figure 2.2: Sample points of adam1

## 2.3.2 Use of Sign Information

In this subsection, we present another technique that is inspired by A. Strzeboński's suggestion, to avoid symbolic computation using the sign information of cells. The quick tests presented in the previous subsection do not work well when the intervals are not sufficiently precise. In such cases sign information may work effectively. A. Strzeboński proposed the usage of sign information described in [75]. The differences between our approach and that of [75] are that our approach requires only the real roots of projection factors and A. Strzeboński's approach does not use the dynamic evaluation technique.

Let $F_{k+1} \subset \mathbb{Q}[x_1, \ldots, x_k, x_{k+1}]$ be a finite set of polynomials, $F_k \subset \mathbb{Q}[x_1, \ldots, x_k]$ an output of a projection operator Proj, and $\mathcal{D}_k$ a CAD of $F_k$. Here we consider lifting a cell $C^{(k)} \in \mathcal{D}_k$ in which $F_k$ is sign invariant. In this step, we have already obtained the signs of $F_k$ at $s \in C^{(k)}$. As $F_k$ is computed by Proj from $F_{k+1}$, we can use this sign information to avoid symbolic computation.

Of course, this technique depends on the choice of projection operator. Here we explain the technique using S. McCallum's projection operator ProjMC from [58], shown in Algorithm 3. The sub-procedures Coeffs$(f, x)$, Discriminant$(f, x)$, and Resultant$(f, g, x)$ return the set of all the coefficients of $f$ w.r.t. $x$, the discriminant of $f$ w.r.t. $x$, and the resultant of $f$ and $g$ w.r.t. $x$, respectively.

**Algorithm 3** ProjMC($F_{k+1}$, $x_{k+1}$)

```
Input: polynomials $F_{k+1} \subset \mathbb{Q}[x_1, \ldots, x_{k+1}]$ $(k \geq 1)$, main variable $x_{k+1}$
Output: projection factors $F_k \subset \mathbb{Q}[x_1, \ldots, x_k]$
  $G \leftarrow \emptyset$
  for $f \in F_{k+1}$ do
    $G \leftarrow G \cup$ Coeffs$(f, x_{k+1})$
    $G \leftarrow G \cup \{$Discriminant$(f, x_{k+1})\}$
  end for
  for $f, g \in F_{k+1}$ $(f \neq g)$ do
    $G \leftarrow G \cup \{$Resultant$(f, g, x_{k+1})\}$
  end for

  return irreducible factors of $G$
```

### Avoidance of Symbolic Zero Determination

As mentioned in Section 2.3.1, to isolate the real roots of $f \in F_{k+1}$ at $s$ in $C^{(k)}$ in the lifting phase, we determine the signs of all the coefficients of the interval polynomial obtained by substituting each isolating interval of $s_i$ for $x_i$ in $f$. Since $F_k$ contains the coefficients of $F_{k+1}$ with respect to $x_{k+1}$ and the signs of $F_k$ at $s$ are already computed, we can reuse them for zero determination in Algorithm 1.

We note that some of the projection operators (e.g., [59, 12]) may not produce all the coefficients of input polynomials.

### Avoidance of Unnecessary Square-free Decomposition

Using sign information of discriminants, we can skip the unnecessary symbolic square-free computation. Moreover, we can do more for quadratic polynomials.

**A Quick Test of Square Freeness**: In Section 2.3.1, we mentioned that we use DE to represent algebraic extensions, which requires the square freeness of defining polynomials. However, employing the numeric real root isolation method we can avoid most of the unnecessary symbolic square-free decomposition applied to square-free polynomials. Still, we might execute an unnecessary square-free decomposition if the precision of the isolating intervals of the sample point is not sufficient. Here we show how we can avoid the symbolic square-free decomposition using the sign of the discriminant of $f(x_1, \ldots, x_k, x_{k+1})$ at $s$.

**Lemma 10.** *Let $f(x_1, \ldots, x_k, x_{k+1}) = \sum_{i=0}^{d} c_i(x_1, \ldots, x_k)x_{k+1}^i$ be a polynomial of degree $d$, $D(x_1, \ldots, x_k)$ the discriminant of $f$ with respect to $x_{k+1}$, and $s$ a point in $\mathbb{R}^k$. The discriminant $D(x_1, \ldots, x_k)$ vanishes at $s$ if and only if $c_d(s)$ and $c_{d-1}(s)$ are equal to zero or $f(s, x_{k+1})$ has a multiple root.*

Since the discriminant $D(x_1, \ldots, x_k)$ of $f(x_1, \ldots, x_k, x_{k+1})$ belongs to $F_k$, we have already obtained the sign of $D(s)$. When $D(s)$ is not equal to zero, we do not need to carry out symbolic square-free decomposition and we thus incorporate this quick test into Algorithm 2. Through this incorporation, we can avoid most of the symbolic square-free decomposition when $f(s, x_{k+1})$ does not have a multiple real root. Thus, DE is a very useful tool for CAD implementation.

**Square-free Decomposition of Quadratic Polynomials**: Using the sign information of the discriminant, we can avoid symbolic square-free decomposition for polynomials of degree two over algebraic extension fields.

**Lemma 11.** *Let $f(x) = c_2x^2 + c_1x + c_0$ be a polynomial ($c_2 \neq 0$). If $f(x)$ has a multiple root, there exists $w \in \mathbb{R}$ such that*

$$f(x) = c_2x^2 + c_1x + c_0 = w(2c_2x + c_1)^2,$$

*where* $\operatorname{sgn}(w) = \operatorname{sgn}(c_2)$.

The function $\operatorname{sgn}(x)$, defined for all real values of $x$, is zero if $x$ is equal to zero, and $x/|x|$ otherwise.

From Lemmas 10 and 11, we have the following lemma.

**Lemma 12.** *Let* $f(x_1, \dots, x_k, x_{k+1}) = \sum_{i=0}^{d} c_i(x_1, \dots, x_k)x_{k+1}^i$ *be a polynomial of degree* $d$*,* $D(x_1, \dots, x_k)$ *the discriminant of* $f$ *with respect to* $x_{k+1}$*, and* $s$ *a point in* $\mathbb{R}^k$*. If the degree of* $f(s, x_{k+1})$ *is two,* $D(s)$ *is equal to zero, and* $d = 2$ *or* $d = 3$*, then up to a positive constant number,* $f(s, x_{k+1})$ *is factorized as*

$$\operatorname{sgn}(c_2(s))(2c_2(s)x_{k+1} + c_1(s))^2. \tag{2.1}$$

Since we are interested only in the signs of projection factors in CAD construction, we can replace the symbolic square-free decomposition of $f(s, x_{k+1})$ with a polynomial in (2.1). We note that the sign of $c_2(s)$ is already known from the sign information. Hence we can avoid the heavy symbolic computation in an algebraic extension field.
**Sign Invariance of Quadratic Polynomials**: We also use the following lemma in the lifting phase.

**Lemma 13.** *Let* $f(x_1, \dots, x_k, x_{k+1})$ *be a polynomial,* $D(x_1, \dots, x_k)$ *the discriminant of* $f$ *with respect to* $x_{k+1}$*, and* $s = (s_1, \dots, s_k)$ *a point in* $\mathbb{R}^k$*. If the degree of* $f(s, x_{k+1})$ *is two and* $D(s)$ *is negative,* $f(s, x_{k+1})$ *is sign definite for all* $x_{k+1}$ *in* $\mathbb{R}$*.*

By applying this lemma in the truth value evaluation, we can avoid lifting more cells. We present the details in Section 2.4.2.

### Avoidance of the Symbolic Equality Check of Algebraic Numbers

Let $f, g \in F_{k+1}$ be projection factors, $C^{(k)} \in \mathcal{D}_k$ a cell, and $s = (s_1, \dots, s_k) \in \mathbb{R}^k$ a sample point for $C^{(k)}$. We denote the set of isolating intervals of the real roots of $f$ and $g$ at $s$ by $L_f$ and $L_g$, respectively. In our implementation, if $I_f \in L_f$ intersects $I_g \in L_g$, then we check whether two isolating intervals stand for the same algebraic number in a symbolic sense using their defining polynomials. Here we show that the sign information of their resultant enables us to avoid symbolic computation.

**Lemma 14.** *Let* $f, g$ *be polynomials in* $F_{k+1}$*, and* $s$ *a point in* $\mathbb{R}^k$*. If the resultant of* $f$ *and* $g$ *with respect to* $x_{k+1}$ *is not equal to zero at* $s$*, then they have no common root.*

This is a well-known fact and very important for CAD computation. From Lemma 14, when their resultant is not equal to zero at $s$, we can isolate the intervals by improving the precision of isolating intervals of $s$. When the resultant of $f$ and $g$ is equal to zero, $I_f$ and $I_g$ give the same algebraic number if three conditions hold:

1. The leading coefficients of $f$ and $g$ in $x_{k+1}$ are not equal to zero at $s$,
2. either $f$ or $g$ has only real roots at $s$ and
3. $J_f$ does not intersect $J_g$ for all $J_f (\neq I_f) \in L_f$ and $J_g (\neq I_g) \in L_g$.

**Remark 15.** *By computing all the complex roots of projection factors, we might avoid more symbolic computations as described in [75].*

## 2.4 Avoidance of Lifting Cells

Reducing the number of lifting cells is a fundamental idea for making CAD construction efficient. Using this idea, G.E. Collins and H. Hong presented partial CAD in [21]. Their partial CAD algorithm improves the efficiency of CAD by evaluating a cell before lifting it to reduce the number of cells. We call this evaluation a trial evaluation, and as a quick test, it checks if the input formula has a constant truth value on a cell and if so, the algorithm skips further construction on the cell.

The following simple example given in [21] illustrates the effect of trial evaluation. Let $\varphi(x_1, x_2) = \exists x_2(\psi_1(x_1) \wedge \psi_2(x_1, x_2))$ be a formula where $\psi_1$ and $\psi_2$ are quantifier free. Now, we consider lifting a cell $C^{(1)} \in \mathcal{D}_1$. We have obtained the signs of the projection factors $F_1$, because $F_1$ is sign-invariant in $C^{(1)}$. Thus we can evaluate the formula $\psi_1$. If $\psi_1$ is false in cell $C^{(1)}$, then the formula $\varphi$ is false for all $x_2$. Thus we do not need to lift $C^{(1)}$.

In this section we consider the "$k$-th lifting step" where we lift a cell $C^{(k)} \in \mathcal{D}_k$. We will further discuss trial evaluation with the help of numerical computation to avoid unnecessary lifting cells $C^{(k)} \in \mathcal{D}_k$.

### 2.4.1 Avoidance of Merging and Sorting

When we solve QE problems using a CAD algorithm, we can avoid lifting the cells whose truth values are determined beforehand. We can apply Algorithm 4 if $x_{k+1}$ is quantified. If the truth value of a cell is determined by a quick test, sorting the real root can be skipped.

**Algorithm 4** LIFTING($C^{(k)}$, $F_{k+1}$)

```
Input: a cell $C^{(k)}$, projection factors $F_{k+1}$
  $s \leftarrow$ a sample point of $C^{(k)}$
  $\mathsf{Q}_{k+1} \leftarrow$ the quantifier of $x_{k+1}$
  for $f \in F_{k+1}$ do
    $L_f \leftarrow$ IntvRealRootIsol($f$, $s$)
    for $J \in L_f$ do
      $T \leftarrow$ trial evaluation at $(s_1, \dots, s_k, J)$ using numeric computation
      if ($T$ = true and $\mathsf{Q}_{k+1} = \exists$) or ($T$ = false and $\mathsf{Q}_{k+1} = \forall$) then
        truth value of $C^{(k)} \leftarrow T$

        return
      end if
    end for
  end for
  Merge the results above and sort the roots
```

## 2.4.2 Trial Evaluation Using the Structure of Formulas

The partial CAD algorithm evaluates only atomic formulae whose levels of polynomials are less than or equal to $k$ in the $k$-th lifting step. By evaluating truth values of all atomic formulae we have a good chance of avoiding unnecessary lifting cells. In this subsection, we explain how we can exploit the structure of a formula to decide truth values for more cells.

A first but simple approach is to use Lemma 13, which is applicable to only quadratic polynomials.

**Example 16.** *(adam2-1 [75]) Let us consider the following QE problem.*

$$\varphi \equiv \forall x \forall y \forall z \; ((x \geq 0 \land y \geq 0 \land h(x, y) < 0) \Rightarrow \psi(x, y, z),$$

*where $h(x, y) = 4(x^2 + y^2) - 1$ and $\psi$ is quantifier free.*

*The discriminant of $h$ with respect to $y$ is $-16(2x - 1)(2x + 1)$. When the discriminant is negative, $h$ is positive for all $y$. Thus $\varphi$ is true for all $y$ and we can avoid lifting cells for $x < -1/2$ and $1/2 < x$.*

### Bounded CAD Construction

We might determine the truth value of a formula in a restricted region specified by the given problem using numeric computation. In this case we do not need to construct a CAD in the whole space. We present an effective approach for CAD constructed in a given restricted connected region $\mathcal{U}_r \subseteq \mathbb{R}^r$, where each subregion $\mathcal{U}_i = \{(x_1, \ldots, x_i) \in \mathbb{R}^i | \exists x_{i+1}(x_1, \ldots, x_i, x_{i+1}) \in \mathcal{U}_{i+1}\}$ for $i = 1, \ldots, r-1$ is also connected. We will call this specially constructed CAD a *bounded CAD* construction.

Now we explain the trial evaluation for bounded CAD. Let $\varphi(x_1, \ldots, x_r)$ be an input formula, and $\mathcal{D}_i$ a CAD of $\mathbb{R}^i$ for $i = 1, \ldots, r$. The trial evaluation in partial CAD uses the following lemma [21].

**Lemma 17.** *Let $C^{(k)}$ be a cell in $\mathcal{D}_k$ of $\mathbb{R}^k$ for $1 \leq k < r$, and $s = (s_1, \ldots, s_k)$ a sample point for $C^{(k)}$. If $\varphi(s_1, \ldots, s_k, x_{k+1}, \ldots, x_r)$ has a constant truth value, then we do not need to lift $C^{(k)}$.*

The trial evaluation deals with only the polynomials such that their levels are less than or equal to $k$ in the $k$-th step in the lifting phase. Since we construct CAD only in $\mathcal{U}_r$, the following lemma is immediate.

**Lemma 18.** *Let $C^{(k)}$ be a cell in $\mathcal{D}_k$ for $1 \leq k < r$, $\mathcal{U}_r$ a connected region, and $s = (s_1, \ldots, s_k)$ a sample point for $C^{(k)}$. If $\varphi(s_1, \ldots, s_k, x_{k+1}, \ldots, x_r)$ has a constant truth value in $\mathcal{U}_r$, then we do not need to lift $C^{(k)}$ in bounded CAD construction in $\mathcal{U}_r$.*

Using Lemma 18 as a quick test, we can recognize truth values for more cells, which improves the efficiency.

In our approach, we try to evaluate all the polynomials in input formulae. Hence, we can avoid lifting more cells, and the computation time might decrease.

Additionally, we can avoid symbolic computation in bounded CAD construction. In Section 2.3.2, we presented a quick test to determine whether two isolating intervals $I_f$, $I_g$ stand for the same algebraic number. However, if $I_f, I_g \not\subseteq \mathcal{U}_{k+1}$, then we can skip the check, because its result is unnecessary for the bounded CAD construction. We note that if $x_{k+1}$ is quantifier free, then this approach has an effect on the solution formula construction phase, because some solution formula construction algorithms require the signs of $F_{k+1}$ for all cells in $\mathcal{D}_{k+1}$.

**Remark 19.** *Since it is difficult to implement bounded CAD construction for general $\mathcal{U}_r$, $\mathcal{U}_r$ should be given as a hyperrectangle and a sample point of a cell should be in $\mathcal{U}_k$ when the boundary of $\mathcal{U}_k$ is contained in the cell.*

### Section Polynomials

We first give a definition of a section polynomial.

**Definition 20.** *For a formula $\varphi$ without universal quantification, a* section polynomial *of $\varphi$ is defined as a polynomial $f$, where $\varphi$ implies $f = 0$.*

QE problems sometimes have section polynomials as projection polynomials. Some projection operators have been proposed for formulae with section polynomials: e.g., [59]. Here we assume that a formula $\varphi$ has two or more section polynomials and does not have a universal quantifier. In other words, we consider that $\varphi$ becomes true only when those polynomials vanish simultaneously.

**Lemma 21.** *Let $\varphi$ be a formula without a universal quantifier. If $f_1$ and $f_2$ are section polynomials of $\varphi$, then so is the resultant of $f_1$ and $f_2$.*

We reformulate this lemma for our purpose.

**Lemma 22.** *Let $\varphi$ be a formula without a universal quantifier, $f_1, f_2 \in F_{k+1}$ section polynomials of $\varphi$, $C^{(k)} \in \mathcal{D}_k$ a cell, and $s \in \mathbb{R}^k$ a sample point for $C^{(k)}$. The resultant of $f_1$ and $f_2$ with respect to $x_{k+1}$ is also a section polynomial and if it is different from zero at $s$, then the truth value of $C^{(k)}$ is false.*

Since the resultant of $f_1$ and $f_2$ belongs to $F_k$, we have already obtained its sign information. Thus, we can avoid lifting cells using this condition without symbolic computation.

## 2.5 Reduction of Projection Factors

Many researchers [41, 58, 12] have worked on reducing the number of projection factors, because it is one of the most effective ways to reduce the number of cells produced in CAD and improve the efficiency of CAD computation. Some have discussed how they can reduce projection factors for a special type of formulae. As examples, formulae with strict inequalities [57, 74] and with equational constraints [59, 60] have been investigated. Here we consider bounded CAD construction as a special type of formulae for which we can reduce the number of projection factors.

### 2.5.1 Projection Operator for Bounded CAD

We do not need to decompose the whole space when we solve QE problems using CAD. From a viewpoint of solving QE problems, there may be unnecessary redundant polynomials among projection factors. We introduce a projection operator specialized to bounded CAD.

Let us consider a short example of how we find unnecessary projection factors in bounded CAD construction. We consider the following QE problem.

$$\varphi \equiv \exists x_2\ (x_1^2 + x_2^2 < 1 \land x_1 \geq 0),$$

where $x_1$ is bounded. We obtain the projection factors using a conventional projection operator:

$$F_1 = \{x_1 + 1, x_1, x_1 - 1\}.$$

Since the formula $\varphi$ is obviously false when $x_1 < 0$, we do not need to construct CAD in $x_1 < 0$. Thus, the projection factor $x_1 + 1$ is unnecessary in solving the QE problem $\varphi$. See Figure 2.3.

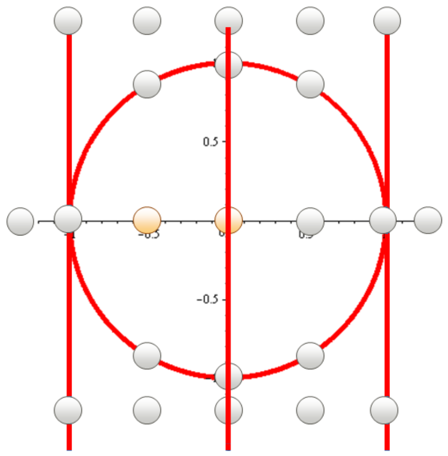

Figure 2.3: Sample points in CAD of $\{x_1^2 + x_2^2 - 1, x_1\}$

The output of a CAD algorithm is a partition of a variable space, where all input polynomials are sign invariant within each cell. Therefore the following lemma holds.

**Lemma 23.** *Let $F \subset \mathbb{Q}[x_1, \ldots, x_r]$ be a finite set of polynomials, $g(x_1, \ldots, x_r)$ a polynomial that does not vanish for any $(x_1, \ldots, x_r) \in \mathbb{R}^r$, and $\mathcal{D}_r$ an output of a CAD algorithm for $F$. Then $g$ is sign invariant in $\mathbb{R}^r$ and thus $F \cup \{g\}$ is sign invariant in each cell $C^{(r)} \in \mathcal{D}_r$.*

Lemma 23 states that a sign-definite polynomial is not needed for CAD construction. The following lemma shows that we might reduce the number of projection factors in our bounded CAD construction.

**Lemma 24.** *Let $F \subset \mathbb{Q}[x_1, \ldots, x_r]$ be a finite set of polynomials, $\mathcal{U}_r \subseteq \mathbb{R}^r$ a connected region, $g(x_1, \ldots, x_r)$ a polynomial that does not vanish for any $(x_1, \ldots, x_r) \in \mathcal{U}_r$, and $\mathcal{D}_r$ an output of a CAD algorithm for $F$. If $C^{(r)} \in \mathcal{D}_r$ is a connected subregion of $\mathcal{U}_r$, then $g$ is sign invariant in $C^{(r)}$ and thus $F \cup \{g\}$ is sign invariant in $C^{(r)}$.*

Using Lemma 24, we propose a new projection operator BProj in Algorithm 5 that can refine any other projection operator Proj by removing unnecessary projection factors in a restricted connected region before applying Proj.

**Algorithm 5** BProj($F_k$, $\mathcal{U}_k$, $x_k$)

**Input:** polynomials $F_k \subset \mathbb{Q}[x_1, \ldots, x_k]$ $(k > 1)$, connected bounded region $\mathcal{U}_k \subseteq \mathbb{R}^k$ given as a semialgebraic set defined by polynomials $B_k$, main variable $x_k$

**Output:** projection factors $F_{k-1} \subset \mathbb{Q}[x_1, \ldots, x_{k-1}]$ for bounded CAD construction in $\mathcal{U}_k$

```
G ← ∅
for $f(x_1, \ldots, x_k) \in F_k$ do
  if $\exists x_1 \cdots \exists x_k ((x_1, \ldots, x_k) \in \mathcal{U}_k \wedge f(x_1, \ldots, x_k) = 0)$ then
    $G \leftarrow G \cup \{f(x_1, \ldots, x_k)\}$
  end if
end for

return Proj($G \cup B_k$, $x_k$)
```

**Remark 25.** *Since a formula might have redundant conditions, we check whether each of the input polynomials of* BProj *is sign definite in $\mathcal{U}_r$.* BProj *does not remove a redundant polynomial in $F_1$. However, unnecessary projection factors of $F_1$ are naturally removed in the base phase.*

**Example 26.** *We consider the following optimization problem [65, p. 31].*

Table 2.1: Number of projection factors of Example 26.

| | | $s$ | | | |
|---|---|---|---|---|---|
| LV | var | 1 | 1/2 | 1/4 | 1/16 |
| 5 | $x_2$ | 4 | 4 | 4 | 4 |
| 4 | $x_1$ | 5 | 5 | 5 | 5 |
| 3 | $\theta_2$ | 8 | 8 | 8 | 8 |
| 2 | $\theta_1$ | 22 | 21 | 21 | 20 |
| 1 | $z$ | 335 | 312 | 309 | 289 |
| total | | 374 | 350 | 347 | 326 |
| time(sec) | | >3600 | 283.2 | 99.1 | 66.4 |

*Problem* $\mathrm{P}(x_0, x_1)$, $\quad 0 \leq \theta_1 \leq s,\ 0 \leq \theta_2 \leq s:$

$$\begin{aligned} \text{Minimize} \quad & f(x_1, x_2) \equiv x_1^3 + 2x_1^2 - 5x_1 + 2x_2^2 - 3x_2 - 6 \\ \text{subject to} \quad & 2x_1 + x_2 \leq 5/2 + \theta_1, \\ & 1/2x_1 + x_2 \leq 3/2 + \theta_2, \\ & x_1 \geq 0, x_2 \geq 0. \end{aligned}$$

*This problem can be formulated as the following first-order formula:*

$$\begin{aligned} \exists x_1 \exists x_2 \quad & (z = f(x_1, x_2) \wedge 2x_1 + x_2 \leq 5/2 + \theta_1 \wedge \\ & 1/2x_1 + x_2 \leq 3/2 + \theta_2 \wedge \\ & x_1 \geq 0 \wedge x_2 \geq 0 \wedge 0 \leq \theta_1 \leq s \wedge 0 \leq \theta_2 \leq s). \end{aligned}$$

*Table 2.1 shows the number of projection factors in each level when s is* 1*,* 1/2*,* 1/4*, and* 1/16*. We solved the QE problems by using a bounded CAD approach with* $\mathcal{U}_5 := 0 \leq \theta_1 \leq s \wedge 0 \leq \theta_2 \leq s$*. We can see that the number of projection factors become smaller when s is small. This result shows that when we can not solve a QE problem by divide the parameter space we might solve it.*

The next example shows another effect of our bounded CAD approach incorporated with McCallum's restricted projection operator.

**Example 27.** *(kimura5) Let us consider the following first-order formula.*

$$\begin{aligned} & \exists x_3 \exists x_4 \exists x_5 \exists x_6\ (f_1 = 0\ \wedge\ f_2 = 0\ \wedge\ f_3 = 0\ \wedge\ f_4 = 0\ \wedge \\ & \quad x_1 \geq x_2 \geq 1\ \wedge\ x_5 > x_3 \geq x_4\ \wedge\ x_5^2 \geq x_6^2\ \wedge\ x_5 \geq 0), \end{aligned}$$

*where*

$$\begin{aligned}
f_1 &= (x_1x_2 + x_1 + x_2)x_3x_4 - (x_1^2x_2 + x_1x_2^2 + x_1x_2),\\
f_2 &= (x_1x_2 + x_1 + x_2)(x_3 + x_4) - (x_1^2 + x_2^2 + x_1^2x_2 + x_1x_2^2 + x_1 + x_2),\\
f_3 &= (x_1^2x_2^2 + x_1^2 + x_2^2)x_5x_6 - (x_1^4x_2^2 + x_1^2x_2^4 + x_1^2x_2^2),\\
f_4 &= (x_1^2x_2^2 + x_1^2 + x_2^2)(x_5 + x_6) - (x_1^4 + x_2^4 + x_1^4x_2^2 + x_1^2x_2^4 + x_1^2 + x_2^2).
\end{aligned}$$

*Now, the formula is evaluated as false outside the region defined by $\mathcal{U}_6$:*

$$\mathcal{U}_6 = (x_1 \geq 1 \wedge x_2 \geq 1 \wedge x_5 \geq 0).$$

*Thus we consider bounded CAD construction in $\mathcal{U}_6$. Note that we can obtain $\mathcal{U}_6$ through numeric computation.*

*At the top level six, there are section polynomials $f_1$ and $f_2$, and we can apply the restricted equational projection operator* ProjEQ *proposed by McCallum [59] at level six. There are no section polynomials of level five in the formula. The resultant of section polynomials $h = h_1^2h_2^2$ is a section polynomial of the formula, where*

$$\begin{aligned}
h_1 &= x_2^2x_1^2 + x_2^2 + x_1^2,\\
h_2 &= -x_2^2x_1^4 + x_1^4x_5^2x_2^2 + x_5^2x_1^4 - x_1^2x_2^4 + x_5^2x_2^4x_1^2 - x_2^2x_1^2 - x_5^4x_2^2x_1^2\\
&\quad -x_5^4x_1^2 + x_5^2x_1^2 + x_5^2x_2^4 - x_5^4x_2^2 + x_5^2x_2^2.
\end{aligned}$$

*We cannot apply the semi-restricted equational projection* ProjEQ$^*$ *[60] at level five because $h$ has two irreducible factors with different level. However, its factor $h_1$ does not vanish except at $x_1 = x_2 = 0$, that is, it never vanish at any point in $\mathcal{U}_6$. Thus the cofactor $h_2$ can be treated as the unique irreducible section polynomial with level five and we can apply* ProjEQ$^*$ *at level five. That is, we can obtain a smaller set of projection factors.*

## 2.6 Procedures of Bounded CAD

Here we explain three procedures for bounded CAD construction presented in Sections 2.4.2 and 2.5.1.

### 2.6.1 Region $\mathcal{U}_r$

As preprocessing of bounded CAD, we compute a region $\mathcal{U}_r$ where the truth value of an input formula $\varphi$ is not trivial. We denote a quantifier-free part of $\varphi$ by $\psi$. We call a region where the truth value of a formula is true or false a *true region* or *false region*, respectively.

In general computing a true region or false region exactly is difficult, but it is often the case that the truth value of a certain subset is easily decided from an atomic formula or two appearing in $\varphi$. We refer to such points that are directly detected true as a *white region.* Similarly we use the term a *black region* for a 'trivially' false subset. Note that detecting white/black regions is important because we can possibly remove some of the projection factors and avoid lifting cells in those regions, which reduces total computing time.

First we compute a white region and a black region according to $\textsc{CompTF}(\psi \wedge (0 = 0), (-\infty, \infty), \emptyset)$. The outputs are $\mathcal{T}_r$ and $\mathcal{F}_r$ expressed by Cartesian products of unions of open intervals. The sub-procedure $\textsc{CompAtom}$ outputs a white region and a black region of a given atomic formula. Let $\mathcal{T}_r, \mathcal{F}_r \subseteq \mathbb{R}^r$ be a white region and a black region of an input formula $\varphi$. Using $\mathcal{T}_r$ and $\mathcal{F}_r$, we easily obtain $\mathcal{U}_r = \mathbb{R}^r \setminus (\mathcal{T}_r \cup \mathcal{F}_r)$.

In our implementation $\mathcal{U}_r$ is expressed by a Cartesian product of closed intervals: i.e., $\mathcal{U}_r = [I_1^{(l)}, I_1^{(u)}] \times \cdots \times [I_r^{(l)}, I_r^{(u)}]$. This is the reason why evaluations are easy in the following procedures of 2.6.2 and 2.6.3.

### 2.6.2 Evaluation of the Sub-QE Problem

Since we express $\mathcal{U}_r$ by a Cartesian product of intervals as Section 2.6.1 it is easy to evaluate the sub-QE problem in Algorithm 5 in the algorithm BPROJ using numerical computation. We simply substitute each interval $[I_i^{(l)}, I_i^{(u)}]$ for $x_i$, and compute a range of values using interval arithmetic. If the range contains zero, we deal with the QE problem as true.

For a necessary projection factor this approach always returns a true value. Therefore we do not lose the correctness as a projection operator.

### 2.6.3 Evaluation of the Truth Value in Region $\mathcal{U}_r$

Here we use the same notation as in lemma 18. In a similar way as 2.6.2, to evaluate the formula $\varphi$ we substitute isolating intervals of $s_i$ for $x_i$ $(i = 1, \dots, k)$ and intervals $[I_j^{(l)}, I_j^{(u)}]$ for $x_j$ $(j = k + 1, \dots, r)$ in each polynomial appearing in $\varphi$, and compute a range of values using interval arithmetic without losing correctness. If $\varphi$ has a constant truth value in $C_k$ over $\mathcal{U}_r$, then we avoid lifting $C_k$.

**Algorithm 6** COMPTF($\psi$, $I_T$, $I_F$)

**Input:** quantifier-free formula $\psi(x_1, \dots, x_r)$, regions $I_T$, $I_F \subseteq \mathbb{R}$

**Output:** white region and black region

```
if $\psi$ is atomic formula then
  return COMPATOM($\psi$, $I_T$, $I_F$)
end if
$\psi \leftarrow$ equivalent conjunction & disjunction formula to $\psi$
$r \leftarrow$ false
if $\psi$ is conjunction then
  $T_0, F_0 \leftarrow (-\infty, +\infty), \emptyset$
  for $p$ in element of conjunction do
    $T$, $F \leftarrow$ COMPTF($p$, $(-\infty, \infty)$, $\emptyset$)
    if error then
      continue
    end if
    $T_0, F_0 \leftarrow T_0 \cap T, F_0 \cup F$
    $r \leftarrow$ true
  end for
else
  $T_0, F_0 \leftarrow \emptyset, (-\infty, +\infty)$
  for $p$ in element of disjunction do
    $T$, $F \leftarrow$ COMPTF($p$, $\emptyset$, $(-\infty, \infty)$)
    if error then
      continue
    end if
    $T_0, F_0 \leftarrow T_0 \cup T, F_0 \cap F$
    $r \leftarrow$ true
  end for
end if
if $r =$ false then
  return error
end if
return $T_0$, $F_0$
```

**Algorithm 7** CompAtom($\psi$, $I_T$, $I_F$)

```
Input: atomic formula $\psi = f(x_1, \dots, x_r)\rho\ 0$, where $\rho \in \{<, \leq, =, \neq\}$, regions $I_T$, $I_F$ $\subseteq \mathbb{R}$
Output: white region and black region
  $T_j \leftarrow I_T$ for $j = 1, \dots, r$
  $F_j \leftarrow I_F$ for $j = 1, \dots, r$
  if $f$ is univariate polynomial in $x_i$ then
    $L_i \leftarrow$ a union of isolating intervals of the real roots of $f$
    $J_i \leftarrow \mathbb{R} \setminus L_i$ (union of open intervals)
    $T_i \leftarrow \{s_i \in J_i \mid f(s_i)\ \rho\ 0\}$
    $F_i \leftarrow \{s_i \in J_i \mid s_i \notin T_i\}$
  else
    for $i = 1$ to $r$ do
      if all coefficients of $x_i$ are rational number except constant term then
        $F \leftarrow$ substitute $(-\infty, \infty)$ for $x_j$ $(j \neq i)$
        transpose $F\ \rho\ 0$ to $g(x_i)\ \rho\ [I^{(l)}, I^{(u)}]$
        update $T_i$, $F_i$ from $g(x_i)\ \rho\ [I^{(l)}, I^{(u)}]$ by same manner as univariate polynomial
      end if
    end for
    if $T_i$, $F_i$ do not update then
      return error
    end if
  end if
  return $T_1 \times \cdots \times T_r$, $F_1 \times \cdots \times F_r$
```

Note that we have already obtained the sign information of the polynomial $f$ such that $n(f) \leq k$, and if $s_k$ is not contained in the interval $[I_k^{(l)}, I_k^{(u)}]$ then a sample point $S$ belongs to $\mathcal{T}_k$ or $\mathcal{F}_k$.

## 2.7 Behavior Analysis

Here we examine and discuss the effects of our improvements according to computational experiments. In more detail, we first analyze the practical behavior of our implementation, and then demonstrate its actual practicality. Since doubly exponential complexity of QE based on CAD in the worst case has been proved, there is no point considering the practicality in a general setting. Instead, we focus on the practicality for actual problems. Here we summarize our improvements. In Section 2.3, we presented ways to use numerical computation (interval arithmetic) to avoid heavy symbolic computation. In Sections 2.4 and 2.5, we presented several improvements to reduce the number of cells.

All the computational experiments were executed on a personal computer with an Intel(R) Core(TM) i3 CPU U330 1.20 GHz and 2.92 GByte memory, and all timing data are given in units of seconds. The mark "-" implies that the program failed to obtain a result because of a shortage of memory. As for the projection phase and solution formula construction phase, we employ the projection operators proposed by [12] and also restricted one by [59] when we can apply it at the top level, and the solution formula construction algorithm shown in [10]. Note that we do not apply the projection operator proposed in [60] even for cases we can apply it.

### 2.7.1 Practical Performance of Our Implementation

In this subsection, we show the practicality of our implementation on actual computation.

**Comparison of Computing Time for Each Improvement**: We first present the timing data of our implementation in Table 2.2. The first column "§2.3" gives the timings of computations with only the improvements presented in Section 2.3. The second column "§2.3–2.4" gives those with only the improvements presented in Sections 2.3 and 2.4. The last column "§2.3–2.5" gives those with all improvements. The problems we deal with here are presented in Section 2.9. The four problems in the first block of the table are taken from [75] concerning with stability study and control design. The second block consists of typical control design problems that are very obstinate for numerical methods of semi-definite programming [63, 54]. Two well-known problems from QE-related papers [55, 10] are given in the third block. The fourth and fifth blocks are QE problems for single–objective optimization [71] and multi–objective optimization [25, 84, 53], respectively. The sixth block is taken from theorem proving problems.

Our improvements presented in Sections 2.4 and 2.5 certainly decreased the computing time in all the problems. The effects of improvements described in Section 2.4 are seen in the problems "mooea" and "wilson", and those described in Section 2.5 are seen in the problems "portfolio" and "kimura5". It will be shown in the next subsection that this good effect on the timing could result from a reduction of the number of cells produced in CAD construction.

**Comparison of Computing Time with Other Implementations**: Comparisons with other implementations are shown in Table 2.3, where SyNRAC represents our implementation with all improvements in Section 2.3 through Section 2.5. QEPCAD B 1.58 [13] was executed with options "+N40000000 +L10000".

SyNRAC was more effective than QEPCAD for relatively large problems that required many symbolic computations over algebraic extension fields. This was due to the avoidance of symbolic computation introduced in Section 2.3 because QEPCAD uses symbolic implementation only. On the other hand, QEPCAD was more effective than SyNRAC for small problems, such as the problems "pl01", "lass", "candj", and "xaxis". We think that these results are due to the overheads of the quick tests.

Mathematica 8.0 [75] solved most of the problems effectively, since it is an implementation with validated numerics. SyNRAC was more effective in the problems derived from optimization; e.g., the problems "portfolio" and "wilson". It seems that in those cases, the bounded CAD improvements presented in Sections 2.4.2 and 2.5.1 certainly reduced the number of cells.

### 2.7.2 Analysis of Our Implementation

In this subsection, using practical computational data, we show that our improvements presented in Sections 2.4 and 2.5 reduced the number of cells, and our improvements presented in Sections 2.3 avoided much of the symbolic computation.

**Number of Cells**: Table 2.4 presents the number of *leaf cells* produced for each problem in each improvement, where a leaf cell is a cell that is not lifted in CAD con-

Table 2.2: Timing data (sec)

| problem | §2.3 | §2.3–2.4 | §2.3–2.5 |
|---|---|---|---|
| adam1 | 0.26 | 0.13 | 0.13 |
| adam2-1 | 9.23 | 1.83 | 1.77 |
| adam2-2 | 4.37 | 4.23 | 4.03 |
| adam3 | 5.82 | 4.82 | 4.84 |
| pl01 | 0.14 | 0.15 | 0.13 |
| lass | 0.34 | 0.28 | 0.30 |
| candj | 0.33 | 0.30 | 0.33 |
| xaxis | 0.59 | 0.52 | 0.55 |
| portfolio | - | 9.16 | 1.42 |
| port-nox3 | 6.67 | 5.98 | 6.07 |
| port-para | - | 11.44 | 11.33 |
| kinoshita | 3.71 | 1.93 | 1.92 |
| mooea | 195.89 | 40.93 | 42.54 |
| wilson | 119.36 | 27.00 | 25.54 |
| lampinen | 31.17 | 17.17 | 16.98 |
| kimura5 | - | 622.82 | 253.89 |
| kimurac | - | 78.66 | 54.35 |

struction. Since the improvements presented in Section 2.3 do not reduce the number of cells, the first column "§2.3" corresponds exactly to the implementation without any improvements in this chapter. For the problems "adam1", "lass", "port-nox3", and "port-para," the number in the second column "§2.3–2.4" is equal to that in the last column "§2.3–2.5". This is because the number of projection factors was not reduced by the improvement presented in Section 2.5.1.

In many problems, our improvements reduced the numbers of cells. We see that the improvements presented in Section 2.5 greatly reduced the numbers of cells in some problems. This result illustrates that the reduction of projection factors is one of the most important techniques in CAD. The numbers of cells were greatly reduced in the problems "mooea" and "wilson" by our improvements presented in Section 2.4, and in the problems "portfolio" and "kimura5" by our improvements presented in Section 2.5.

Table 2.3: Timing data (sec)

| problem | SyNRAC | Mathematica 8.0 | QEPCAD B 1.58 |
|---|---:|---:|---:|
| adam1 | 0.13 | 0.88 | 0.95 |
| adam2-1 | 1.77 | 2.75 | 326 |
| adam2-2 | 4.03 | 4.15 | 806 |
| adam3 | 4.84 | 2.31 | - |
| pl01 | 0.13 | 0.19 | 0.08 |
| lass | 0.30 | 0.43 | 0.10 |
| candj | 0.33 | 0.32 | 0.11 |
| xaxis | 0.55 | 0.29 | 0.18 |
| portfolio | 1.42 | 272 | 524 |
| port-nox3 | 6.07 | 52.55 | 9.62 |
| port-para | 1.33 | 103 | 128 |
| kinoshita | 1.92 | 6.54 | - |
| mooea | 42.54 | 9.74 | - |
| wilson | 25.54 | 28.92 | - |
| lampinen | 16.98 | 13.94 | 239.48 |
| kimura5 | 253.89 | 9.84 | >3600 |
| kimurac | 54.35 | >3600 | - |

Figure 2.4 shows the relation between computing time and the number of cells. From Figure 2.4 and comparing Table 2.2 with Table 2.4, we see that the reduction in the number of cells makes CAD construction efficient.

**Number of Projection Factors**: Table 2.5 presents the number of projection factors. The labels "original", "§2.5" and "removed" stand for an implementation without any improvements, an implementation with the projection operator for bounded CAD construction presented in Section 2.5.1 and the number of the projection factors removed by Algorithm 5, respectively.

Our bounded CAD reduced the number of projection factors in some problems; e.g., problems "adam2-1", "portfolio", and "kimura5".

The problem "portfolio" is a special case where bounded CAD worked very well in reducing the number of projection factors. It has a redundant constraint $x_3 \geq 0$. Interestingly, there were fewer projection factors for the problem "portfolio" than for

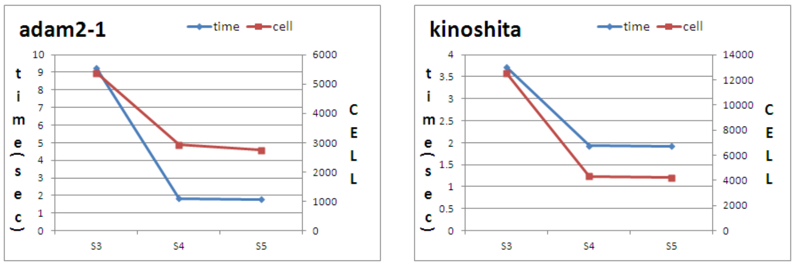

Figure 2.4: Relation between computing time and the number of cells

the problem “port-nox3”, which does not have the redundant constraint.

**Avoidance of Symbolic Computation**: Table 2.6 tells us the numbers of occurrences of important *events* in our implementation. The first three columns “zero test” give the information of the symbolic zero test. The second column “done” gives the number of executed symbolic zero tests, or the number of calls of the function SymbolicZeroChk, the first column “worked” gives the number of cases in which SymbolicZeroChk returns a true value, and the third column “QT” gives the number of executed the quick test presented in §2.3.1. The next four columns give the information of symbolic square-free decomposition. The sixth column “done” gives the number of calls of symbolic square-free decomposition SymbolicSqfr, the fifty column “worked” gives the number of cases that the input of SymbolicSqfr has a multiple root, the fourth column “deg2” gives the number of avoidances of symbolic computation over algebraic extension fields using Lemma 12 for the quadratic polynomials, and the seventh column “QT” gives calls of IntvRealRootIsol presented in §2.3.1. The final column “lift” gives the number of lifting cells.

Table 2.6 shows the effect of our improvements presented in Section 2.3. We avoided most unnecessary symbolic zero tests and square-free decomposition. Unnecessary symbolic computation was executed only in the problem “kimurac”. Most square-free computation was applied to second-degree polynomials.

Moreover, there were far fewer calls of symbolic computation than lifting cells, because we avoided many symbolic computations by reusing the sign information presented in Section 2.3.2.

**Comparison of the Number of Cells with Other Implementations**: Finally, we compare the number of cells produced by SyNRAC with the number produced by other implementation. Table 2.7 gives the number of cells. All data for Mathematica were not obtained.

Table 2.4: Number of cells (leaf)

| problem | §2.3 | §2.3–2.4 | §2.3–2.5 |
|---|---:|---:|---:|
| adam1 | 49 | 21 | 21 |
| adam2-1 | 5737 | 2929 | 2751 |
| adam2-2 | 11173 | 10527 | 9677 |
| adam3 | 10267 | 10153 | 9997 |
| pl01 | 203 | 179 | 121 |
| lass | 537 | 307 | 307 |
| candj | 549 | 549 | 547 |
| xaxis | 2627 | 2627 | 2553 |
| portfolio | - | 37921 | 5065 |
| port-nox3 | 20001 | 18537 | 18537 |
| port-para | - | 43757 | 43757 |
| kinoshita | 12495 | 4335 | 4221 |
| mooea | 2436623 | 472069 | 472531 |
| wilson | 1118095 | 202969 | 202969 |
| lampinen | 129677 | 69663 | 69663 |
| kimura5 | - | 71355 | 27035 |
| kimurac | - | 12691 | 11607 |

SyNRAC produced fewer cells than QEPCAD. This affected the timing data.

### 2.7.3 Statistical Data

We present other statistical data of our implementation with all improvements presented in this chapter.

**Computing Time in Each Phase**: First, we compare the computing time in each phase.

Table 2.8 and Figure 2.5 give the rates of computing time in each phase. The labels "proj", "base", "lift", and "sfc" respectively indicate the projection phase, base phase, and lifting phase for CAD, and the phase of constructing the solution formula, which is the final step in QE based on CAD.

In most of the problems, the projection phase and lifting phase required much com-

Table 2.5: Numbers of projection factors

| problem | original | §2.5 | removed |
|---|---:|---:|---:|
| adam1 | 8 | 8 | 0 |
| adam2-1 | 75 | 68 | 3 |
| adam2-2 | 93 | 83 | 1 |
| adam3 | 43 | 41 | 2 |
| pl01 | 12 | 10 | 1 |
| lass | 21 | 21 | 0 |
| candj | 19 | 17 | 1 |
| xaxis | 30 | 28 | 2 |
| portfolio | 183 | 68 | 2 |
| port-nox3 | 76 | 76 | 0 |
| port-para | 124 | 124 | 0 |
| kinoshita | 55 | 51 | 1 |
| mooea | 255 | 255 | 0 |
| wilson | 310 | 310 | 0 |
| lampinen | 73 | 73 | 0 |
| kimura5 | 324 | 147 | 12 |
| kimurac | 235 | 168 | 8 |

puting time. The problems “wilson” and “lampinen” consumed much time in the phase of constructing the solution formula, as their sets of projection factors did not contain all the polynomials needed for the solution formula. Our SyNRAC implementation is not so efficient for constructing solution formulae, and this must be addressed in our future work.

### Comparison of Levels

We present statistical data for each level.

**Computing Time at Each Level**: Table 2.9 presents the computing time at each level of the projection phase and lifting phase. The first column “lv” gives the level. The next two columns “proj” and the last four columns “lift” give the information of the projection phase and lifting phase, respectively. The labels “time”, “rate”, “count”,

Table 2.6: Numbers of occurrences: §2.3–2.5

| | zero test | | | square-free | | | | |
|---|---|---|---|---|---|---|---|---|
| problem | worked | done | QT | deg2 | worked | done | QT | lift |
| adam1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| adam2-1 | 0 | 0 | 132 | 2 | 0 | 0 | 459 | 265 |
| adam2-2 | 41 | 41 | 582 | 34 | 36 | 36 | 2762 | 1259 |
| adam3 | 483 | 483 | 1047 | 21 | 99 | 99 | 1481 | 782 |
| pl01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 |
| lass | 6 | 6 | 4 | 0 | 2 | 2 | 5 | 25 |
| candj | 6 | 6 | 29 | 0 | 1 | 1 | 49 | 76 |
| xaxis | 32 | 32 | 35 | 14 | 14 | 14 | 87 | 302 |
| portfolio | 11 | 11 | 475 | 27 | 27 | 27 | 541 | 621 |
| port-nox3 | 128 | 128 | 5355 | 580 | 580 | 580 | 6042 | 7354 |
| port-para | 32 | 32 | 3750 | 499 | 499 | 499 | 5555 | 8514 |
| kinoshita | 108 | 108 | 769 | 46 | 48 | 48 | 1134 | 1235 |
| mooea | 3041 | 3041 | 16687 | 582 | 594 | 594 | 22092 | 70708 |
| wilson | 0 | 0 | 6918 | 297 | 297 | 297 | 11221 | 35157 |
| lampinen | 820 | 820 | 42908 | 72 | 174 | 174 | 50950 | 21252 |
| kimura5 | 424 | 437 | 5331 | 1 | 1 | 1 | 7635 | 14222 |
| kimurac | 331 | 321 | 1839 | 26 | 26 | 26 | 2619 | 5694 |

and “mean” indicate the computing time at each level, the rate of computing time, the number of lifting cells, and the mean time of computing one cell, respectively.

The computing time at the lower level became longer in the projection phase as the number of projection factors increased, such as in the problems “portfolio” and “port-para”. In the lifting phase, the computing time and the mean time of computing one cell were shorter at the high level, even if the extension degree was high, because the number of projection factors and that of cells produced by symbolic computation were smaller.

**Types of Cells at Each Level**: Table 2.10 shows the information of section cells at each level. Labels “lv”, “q”, “rational”, “symbolic”, “numeric”, and “total” indicate the level, quantifier, number of cells constructed over the rational number field, number of cells constructed over an algebraic extension field with symbolic computa-

Table 2.7: Numbers of cells

| problem | SyNRAC | QEPCAD B 1.58 |
|---|---|---|
| adam1 | 21 | 58 |
| adam2-1 | 2751 | 6835 |
| adam2-2 | 9677 | 11653 |
| pl01 | 121 | 225 |
| lass | 307 | 1328 |
| candj | 547 | 685 |
| xaxis | 2553 | 3029 |
| portfolio | 5065 | 763190 |
| port-nox3 | 18537 | 76470 |
| port-para | 43757 | 520953 |
| lampinen | 69663 | 410144 |

tion, number of cells constructed over an algebraic extension field without symbolic computation, and number of section cells, respectively.

These tables show that our improvement avoided much symbolic computation in most of the problems. In particular, our improvement required few symbolic computations at the highest level. We believe that this is one of the reasons why the computing time was shorter at a higher level in Table 2.9.

**Numbers of Projection Factors at Each Level**: Table 2.11 presents the number of projection factors at each level. The lower the level was, the more projection factors there were. This is another reason why the computing time was shorter at a higher level in Table 2.9.

Table 2.9: Timing data at each level

| | | proj | | lift | | | |
|---|---|---|---|---|---|---|---|
| problem | lv | time | rate | time | rate | count | mean |
| adam2-1 | 2 | 0.76 | 60.28 | 0.06 | 12.94 | 234 | 0.00026 |
| adam2-1 | 1 | 0.50 | 39.72 | 0.40 | 87.06 | 31 | 0.01298 |
| adam2-2 | 2 | 1.03 | 56.98 | 0.51 | 24.43 | 1160 | 0.00044 |
| adam2-2 | 1 | 0.78 | 43.02 | 1.59 | 75.57 | 99 | 0.01609 |

Table 2.9: Timing data at each level

| | | proj | | lift | | | |
|---|---|---|---|---|---|---|---|
| problem | lv | time | rate | time | rate | count | mean |
| adam3 | 3 | 0.32 | 26.12 | 1.40 | 39.58 | 495 | 0.00283 |
| adam3 | 2 | 0.32 | 26.36 | 1.14 | 32.07 | 244 | 0.00465 |
| adam3 | 1 | 0.57 | 47.52 | 1.00 | 28.35 | 43 | 0.02335 |
| pl01 | 2 | 0.06 | 83.46 | 0.01 | 14.50 | 7 | 0.00107 |
| pl01 | 1 | 0.01 | 16.54 | 0.04 | 85.50 | 5 | 0.00882 |
| lass | 3 | 0.00 | 1.02 | 0.00 | 0.29 | 5 | 0.00012 |
| lass | 2 | 0.07 | 71.29 | 0.09 | 45.24 | 14 | 0.00653 |
| lass | 1 | 0.03 | 27.69 | 0.11 | 54.46 | 6 | 0.01835 |
| candj | 2 | 0.15 | 81.15 | 0.01 | 4.18 | 66 | 0.00009 |
| candj | 1 | 0.04 | 18.85 | 0.13 | 95.82 | 10 | 0.01324 |
| xaxis | 4 | 0.15 | 47.88 | 0.05 | 22.04 | 152 | 0.00034 |
| xaxis | 3 | 0.09 | 29.29 | 0.08 | 34.55 | 112 | 0.00073 |
| xaxis | 2 | 0.04 | 14.08 | 0.09 | 36.20 | 32 | 0.00269 |
| xaxis | 1 | 0.03 | 8.75 | 0.02 | 7.22 | 6 | 0.00286 |
| portfolio | 3 | 0.01 | 4.07 | 0.14 | 11.40 | 305 | 0.00047 |
| portfolio | 2 | 0.04 | 31.51 | 0.47 | 37.64 | 226 | 0.00208 |
| portfolio | 1 | 0.09 | 64.42 | 0.64 | 50.96 | 90 | 0.00708 |
| port-nox3 | 3 | 0.01 | 4.05 | 1.90 | 32.61 | 5824 | 0.00033 |
| port-nox3 | 2 | 0.06 | 29.27 | 2.90 | 49.77 | 1401 | 0.00207 |
| port-nox3 | 1 | 0.13 | 66.68 | 1.03 | 17.62 | 129 | 0.00795 |
| port-para | 3 | 0.01 | 2.65 | 2.36 | 21.73 | 5598 | 0.00042 |
| port-para | 2 | 0.08 | 30.72 | 8.26 | 76.04 | 2830 | 0.00292 |
| port-para | 1 | 0.17 | 66.62 | 0.24 | 2.23 | 86 | 0.00281 |
| kinoshita | 3 | 0.04 | 4.49 | 0.03 | 4.49 | 437 | 0.00006 |
| kinoshita | 2 | 0.33 | 32.51 | 0.20 | 32.51 | 230 | 0.00086 |
| kinoshita | 1 | 0.63 | 63.00 | 0.39 | 63.00 | 54 | 0.00713 |
| mooea | 4 | 0.01 | 1.85 | 2.68 | 6.60 | 46604 | 0.00006 |
| mooea | 3 | 0.03 | 6.46 | 32.07 | 78.92 | 17544 | 0.00183 |
| mooea | 2 | 0.06 | 15.27 | 4.00 | 9.84 | 6395 | 0.00063 |

Table 2.9: Timing data at each level

| | | proj | | lift | | | |
|---|---|---|---|---|---|---|---|
| problem | lv | time | rate | time | rate | count | mean |
| mooea | 1 | 0.32 | 76.42 | 1.89 | 4.64 | 165 | 0.01143 |
| wilson | 3 | 0.01 | 2.17 | 1.74 | 7.46 | 26585 | 0.00007 |
| wilson | 2 | 0.22 | 41.10 | 18.91 | 81.09 | 8347 | 0.00227 |
| wilson | 1 | 0.30 | 56.73 | 2.67 | 11.45 | 226 | 0.01182 |
| lampinen | 3 | 0.03 | 10.25 | 1.46 | 9.20 | 17908 | 0.00008 |
| lampinen | 2 | 0.13 | 48.92 | 13.67 | 86.04 | 3225 | 0.00424 |
| lampinen | 1 | 0.11 | 40.83 | 0.76 | 4.77 | 119 | 0.00637 |
| kimura5 | 5 | 0.42 | 3.20 | 0.01 | 0.00 | 846 | 0.00001 |
| kimura5 | 4 | 0.46 | 3.56 | 1.66 | 0.70 | 410 | 0.00406 |
| kimura5 | 3 | 0.09 | 0.67 | 0.26 | 0.11 | 12490 | 0.00002 |
| kimura5 | 2 | 0.59 | 4.51 | 3.75 | 1.57 | 410 | 0.00915 |
| kimura5 | 1 | 11.45 | 88.07 | 232.75 | 97.62 | 66 | 3.52651 |
| kimurac | 4 | 0.38 | 4.24 | 2.90 | 6.56 | 44 | 0.06583 |
| kimurac | 3 | 0.02 | 0.25 | 0.07 | 0.17 | 5282 | 0.00001 |
| kimurac | 2 | 0.66 | 7.40 | 2.12 | 4.80 | 324 | 0.00654 |
| kimurac | 1 | 7.87 | 88.11 | 39.08 | 88.48 | 44 | 0.88815 |

Table 2.10: Number of cells (nodes): §2.3–2.5

| problem | lv | q | rational | symbolic | numeric | total |
|---|---|---|---|---|---|---|
| adam1 | 2 | $\forall$ | 0 | 0 | 0 | 0 |
| adam1 | 1 | $\forall$ | 10 | 0 | 0 | 10 |
| adam1 | T | | 10 | 0 | 0 | 10 |
| adam2-1 | 3 | $\forall$ | 325 | 0 | 347 | 672 |
| adam2-1 | 2 | $\forall$ | 357 | 0 | 264 | 621 |
| adam2-1 | 1 | $\forall$ | 82 | 0 | 0 | 82 |
| adam2-1 | T | | 764 | 0 | 611 | 1375 |

Table 2.10: Number of cells (nodes): §2.3–2.5

| problem | lv | q | rational | symbolic | numeric | total |
|---|---|---|---|---|---|---|
| adam2-2 | 3 | ∀ | 1263 | 0 | 1307 | 2570 |
| adam2-2 | 2 | ∀ | 1226 | 69 | 866 | 2161 |
| adam2-2 | 1 | ∀ | 107 | 0 | 0 | 107 |
| adam2-2 | T | | 2596 | 69 | 2173 | 4838 |
| adam3 | 4 | ∀ | 660 | 348 | 1344 | 2352 |
| adam3 | 3 | ∃ | 815 | 179 | 866 | 1860 |
| adam3 | 2 | ∃ | 470 | 66 | 221 | 757 |
| adam3 | 1 | * | 29 | 0 | 0 | 29 |
| adam3 | T | | 1974 | 593 | 2431 | 4998 |
| pl01 | 3 | ∀ | 23 | 0 | 0 | 23 |
| pl01 | 2 | ∀ | 35 | 0 | 0 | 35 |
| pl01 | 1 | * | 2 | 0 | 0 | 2 |
| pl01 | T | | 60 | 0 | 0 | 60 |
| lass | 4 | ∀ | 30 | 0 | 0 | 30 |
| lass | 3 | ∀ | 68 | 10 | 0 | 78 |
| lass | 2 | ∀ | 42 | 0 | 0 | 42 |
| lass | 1 | * | 3 | 0 | 0 | 3 |
| lass | T | | 143 | 10 | 0 | 153 |
| candj | 3 | ∃ | 149 | 0 | 50 | 199 |
| candj | 2 | * | 63 | 3 | 0 | 66 |
| candj | 1 | * | 8 | 0 | 0 | 8 |
| candj | T | | 220 | 3 | 50 | 273 |
| xaxis | 5 | ∀ | 468 | 10 | 108 | 586 |
| xaxis | 4 | ∀ | 410 | 0 | 94 | 504 |
| xaxis | 3 | * | 128 | 0 | 16 | 144 |
| xaxis | 2 | * | 35 | 0 | 0 | 35 |
| xaxis | 1 | * | 7 | 0 | 0 | 7 |
| xaxis | T | | 1048 | 0 | 218 | 1276 |
| portfolio | 4 | ∃ | 5 | 0 | 0 | 5 |
| portfolio | 3 | ∃ | 743 | 51 | 406 | 1200 |

Table 2.10: Number of cells (nodes): §2.3–2.5

| problem | lv | q | rational | symbolic | numeric | total |
|---|---|---|---|---|---|---|
| portfolio | 2 | ∃ | 1062 | 0 | 0 | 210 |
| portfolio | 1 | * | 7 | 0 | 0 | 7 |
| portfolio | T | | 7 | 0 | 0 | 7 |
| port-nox3 | 4 | ∃ | 5 | 0 | 0 | 5 |
| port-nox3 | 3 | ∃ | 4106 | 460 | 2476 | 7042 |
| port-nox3 | 2 | ∃ | 1799 | 0 | 358 | 2157 |
| port-nox3 | 1 | * | 64 | 0 | 0 | 64 |
| port-nox3 | T | | 5974 | 460 | 2834 | 9268 |
| port-para | 4 | ∃ | 509 | 7 | 174 | 690 |
| port-para | 3 | ∃ | 14569 | 46 | 5048 | 19663 |
| port-para | 2 | * | 1047 | 0 | 368 | 1415 |
| port-para | 1 | * | 110 | 0 | 0 | 110 |
| port-para | T | | 16235 | 53 | 5590 | 21878 |
| kinoshita | 4 | ∃ | 0 | 2 | 6 | 8 |
| kinoshita | 3 | ∃ | 531 | 113 | 741 | 1385 |
| kinoshita | 2 | ∃ | 369 | 71 | 237 | 677 |
| kinoshita | 1 | * | 40 | 0 | 0 | 40 |
| kinoshita | T | | 940 | 186 | 984 | 2110 |
| mooea | 5 | ∃ | 36 | 0 | 373 | 409 |
| mooea | 4 | ∃ | 82279 | 0 | 68673 | 150952 |
| mooea | 3 | ∃ | 43028 | 12 | 35063 | 78103 |
| mooea | 2 | * | 4584 | 0 | 2036 | 6619 |
| mooea | 1 | * | 182 | 0 | 0 | 182 |
| mooea | T | | 130109 | 12 | 106144 | 236265 |
| wilson | 4 | ∃ | 56 | 0 | 224 | 280 |
| wilson | 3 | ∃ | 50719 | 0 | 43375 | 94094 |
| wilson | 2 | * | 4810 | 0 | 2034 | 6844 |
| wilson | 1 | * | 266 | 0 | 0 | 266 |
| wilson | T | | 55851 | 0 | 45633 | 101484 |
| lampinen | 4 | ∃ | 40 | 0 | 524 | 564 |

Table 2.10: Number of cells (nodes): §2.3–2.5

| problem | lv | q | rational | symbolic | numeric | total |
|---|---|---|---|---|---|---|
| lampinen | 3 | ∃ | 13570 | 959 | 17397 | 31926 |
| lampinen | 2 | * | 1228 | 32 | 995 | 2255 |
| lampinen | 1 | * | 73 | 0 | 0 | 73 |
| lampinen | T | | 14911 | 991 | 18279 | 34818 |
| kimura5 | 6 | ∃ | 0 | 0 | 0 | 0 |
| kimura5 | 5 | ∃ | 764 | 0 | 3948 | 4712 |
| kimura5 | 4 | ∃ | 133 | 0 | 686 | 819 |
| kimura5 | 3 | ∃ | 2459 | 805 | 2981 | 6245 |
| kimura5 | 2 | * | 844 | 341 | 393 | 1578 |
| kimura5 | 1 | * | 163 | 0 | 0 | 163 |
| kimura5 | T | | 4363 | 1146 | 8008 | 13517 |
| kimurac | 5 | ∃ | 2 | 0 | 0 | 2 |
| kimurac | 4 | ∃ | 47 | 0 | 40 | 87 |
| kimurac | 3 | ∃ | 1809 | 59 | 2722 | 4590 |
| kimurac | 2 | * | 553 | 332 | 110 | 995 |
| kimurac | 1 | * | 129 | 0 | 0 | 129 |
| kimurac | T | | 2540 | 391 | 2872 | 5803 |

## 2.8 Conclusion

We have proposed our strategy using quick tests aimed at improving the efficiency of the Collins QE algorithm based on CAD and implemented all our improvements on a Maple package called SyNRAC. We consider there to be two most important targets in making QE based on CAD more efficient. One is to reduce symbolic computation in CAD construction. The other is to reduce the number of cells produced during CAD. Six quick tests in total were introduced to achieve these goals. These tests were proposed in Sections 2.3 to 2.5. Quick tests using numerical computation were explained in Section 2.3, quick tests before lifting a cell in Section 2.4, and quick tests to produce fewer projection factors in Section 2.5.

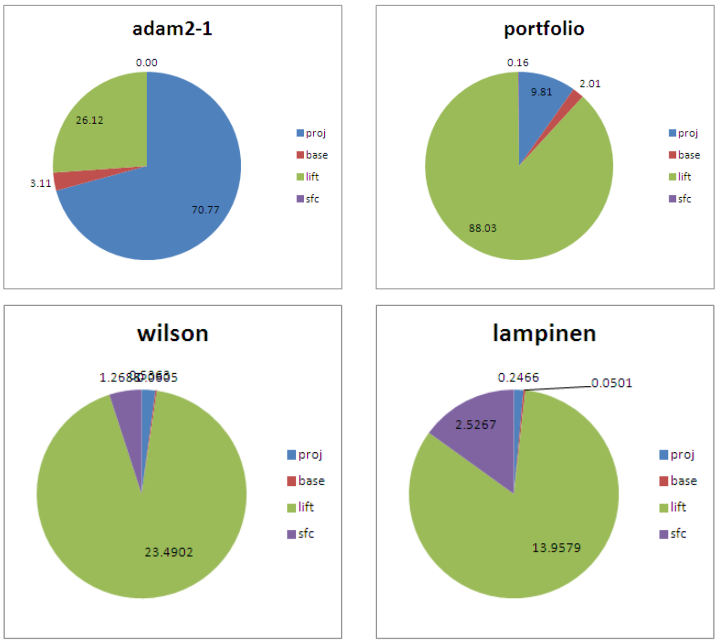

Figure 2.5: Rate of computing time in each phase

The effectiveness of our quick tests was verified using statistical data taken from many example problems. Our implementation showed that numerical quick tests can avoid heavy symbolic counterparts. Quick tests before lifting a cell help prevent our CAD algorithm from making unnecessary stacks. These tests save much more time at a higher level in lifting. Quick tests in the projection phase not only saved computing time in that phase itself, but also in reduced the number of cells produced in lifting.

The methods proposed in the present chapter will be applied to other algorithms in real algebraic geometry. Ideas behind bounded CAD, for example, can be used to improve QE by *virtual term substitution* [55, 81, 82].

We will study other methods to improve our algorithm. Experimental results showed that the solution formula construction phase dominates the computing time when the set of projection factors does not contain all the polynomials appearing in the solution formula. We are eager to employ methods other than 'extended Tarski formulae'. Furthermore it seems promising to combine SNCAD with the ideas of discriminant variety [69] in solving large practical problems.

Table 2.8: Rate of timing data in each phase

| problem | proj | base | lift | sfc |
|---|---|---|---|---|
| adam1 | 90.73 | 9.22 | 0.00 | 0.05 |
| adam2-1 | 70.77 | 3.11 | 26.12 | 0.00 |
| adam2-2 | 45.00 | 2.69 | 52.31 | 0.00 |
| adam3 | 24.92 | 1.89 | 73.17 | 0.02 |
| pl01 | 59.61 | 0.56 | 39.68 | 0.15 |
| lass | 32.39 | 0.14 | 67.40 | 0.07 |
| candj | 57.22 | 0.49 | 41.86 | 0.43 |
| xaxis | 56.18 | 0.17 | 43.16 | 0.49 |
| portfolio | 9.81 | 2.01 | 88.02 | 0.16 |
| port-nox3 | 3.19 | 0.85 | 95.89 | 0.07 |
| port-para | 2.27 | 0.39 | 95.87 | 1.47 |
| kinoshita | 31.85 | 4.58 | 63.50 | 0.07 |
| mooea | 1.00 | 0.15 | 97.11 | 1.74 |
| wilson | 2.12 | 0.24 | 92.64 | 5.00 |
| lampinen | 1.47 | 0.30 | 83.18 | 15.05 |
| kimura5 | 5.19 | 0.71 | 94.05 | 0.04 |
| kimurac | 16.77 | 0.72 | 81.62 | 0.28 |

## 2.9 Examples

adam1 [75] Stability of Dormand-Prince fifth-order embedded seven-stage method (Example 4.4 from Hong et al. (1997))

$$\begin{aligned}\forall x \forall y\ ((x < 0 \wedge x^2 + y^2 < \frac{99438}{100000})\\ \Rightarrow R(x+jy)R(x-jy) < 1),\end{aligned}$$

where

$$R(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \frac{z^5}{120} + \frac{z^6}{600}.$$

adam2 [75] Stability of a six-point upwind-based second-order accurate scheme for approximating a two-dimensional advection equation (Example 5.4 from Hong et

Table 2.11: Number of projection factors: §2.3–2.5

| | Level | | | | | | |
|---|---|---|---|---|---|---|---|
| problem | 6 | 5 | 4 | 3 | 2 | 1 | total |
| adam1 | | | | | 2 | 6 | 8 |
| adam2-1 | | | | 3 | 12 | 53 | 68 |
| adam2-2 | | | | 4 | 16 | 63 | 83 |
| adam3 | | | 2 | 5 | 9 | 25 | 41 |
| pl01 | | | | 3 | 5 | 2 | 10 |
| lass | | | 6 | 6 | 6 | 3 | 21 |
| candj | | | | 4 | 6 | 7 | 17 |
| xaxis | | 2 | 5 | 6 | 8 | 7 | 28 |
| portfolio | | | 3 | 4 | 10 | 51 | 68 |
| port-nox3 | | | 3 | 4 | 11 | 58 | 76 |
| port-para | | | 4 | 5 | 17 | 98 | 124 |
| kinoshita | | | 5 | 5 | 9 | 32 | 51 |
| mooea | | 4 | 7 | 13 | 39 | 192 | 255 |
| wilson | | | 4 | 9 | 27 | 270 | 310 |
| lampinen | | | 4 | 7 | 14 | 59 | 84 |
| kimura5 | 4 | 6 | 3 | 8 | 17 | 109 | 147 |
| kimurac | | 4 | 3 | 8 | 17 | 136 | 168 |

al. (1997))

$$\begin{aligned} 2-1 \quad & \forall\alpha\forall\beta\forall C_2\ ((\alpha \geq 0 \wedge \beta \geq 0 \wedge 4(\alpha^2+\beta^2) < 1) \\ & \qquad \Rightarrow (B \leq 0 \vee D \leq 0)), \\ 2-2 \quad & \forall\alpha\forall\beta\forall C_2\ ((0 \leq \alpha \leq 1 \wedge 0 \leq \beta \leq 1) \\ & \qquad \Rightarrow A \leq 0 \wedge C \leq 0 \wedge (B \leq 0 \vee D \leq 0)), \end{aligned}$$

where

$$\begin{aligned}
A &= C_2^4(\alpha-\beta+1)(\alpha-\beta-1)(\alpha-\beta)^2,\\
B &= 2C_2^4\beta(3\alpha^2\beta-2\alpha^2-2\alpha\beta^2+\alpha+\beta^3-\beta)\\
&\quad +4C_2^3\alpha\beta(\alpha^2-\alpha+\beta^2-\beta)\\
&\quad +2C_2^2\alpha(\alpha^3-2\alpha^2\beta+3\alpha\beta^2-\alpha-2\beta^2+\beta),\\
C &= C_2^4\beta^2(\beta^2-1)+4C_2^3\alpha\beta^2(\beta-1)+\alpha^2(\alpha^2-1)\\
&\quad +2C_2^2\alpha\beta(3\alpha\beta-2\alpha-2\beta+1)+4C_2\alpha^2\beta(\alpha-1),\\
D &= C_2^2R+2C_2S+T,\\
R &= 8\alpha^2\beta^2-12\alpha^2\beta+5\alpha^2-8\alpha\beta^3+8\alpha\beta^2+2\alpha\beta\\
&\quad -4\alpha+4\beta^4-4\beta^3-3\beta^2+4\beta,\\
S &= 4\alpha^3\beta-2\alpha^3-4\alpha^2\beta^2-2\alpha^2\beta+\alpha^2+4\alpha\beta^3\\
&\quad -2\alpha\beta^2+2\alpha\beta-2\beta^3+\beta^2,\\
T &= 4\alpha^4-8\alpha^3\beta-4\alpha^3+8\alpha^2\beta^2+8\alpha^2\beta-3\alpha^2\\
&\quad -12\alpha\beta^2+2\alpha\beta+4\alpha+5\beta^2-4\beta.
\end{aligned}$$

adam3 [75] Robust multi-objective feedback design (Example 4.2 from Dorato et al. (1997))

Find the set of $\frac{n}{d}$ satisfying:

$$\begin{aligned}
&\exists q_1\exists q_2\forall w\ (q_1>1\ \wedge\ q_2>0\ \wedge\ \tfrac{n}{d}>0\ \wedge\\
&\quad (\tfrac{n}{d}-q_1^2)w^4+(\tfrac{n}{d}((q_1+1)^2-2q_2)-(q_1^2+q_2^2))w^2+(\tfrac{n}{d}-1)q_2^2\ge 0\ \wedge\\
&\quad (\tfrac{n}{d}-q_1^2)w^4+(\tfrac{n}{d}((q_1-1)^2-2q_2)-(q_1^2+q_2^2))w^2+(\tfrac{n}{d}-1)q_2^2\ge 0).
\end{aligned}$$

pl01 [63]

$$\begin{aligned}
&\forall t_1\forall t_2\ ((-1\le t_1\le 1\wedge -1\le t_2\le 1)\\
&\quad\Rightarrow\ f\le t_1^2t_2^4+t_1^4t_2^2+1-3t_1^2t_2^2)
\end{aligned}$$

lass [54]

$$\begin{aligned}
&\forall t_1\forall t_2\forall\rho\ ((0\le\rho<1\ \wedge\ -\rho\le t_1\le\rho\ \wedge -\rho\le t_2\le\rho)\\
&\quad\Rightarrow\ f\le t_1^2t_2^4+t_1^4t_2^2+1-t_1^2t_2^2)
\end{aligned}$$

candj: Collins and Johnson 1989b [55]

$$\begin{aligned}
&\exists r\ (0<r<1\ \wedge\ b>0\ \wedge\ a\ge 1/2\ \wedge\\
&\quad 3a^2r+3b^2r-2ar-a^2-b^2<0\\
&\quad 3a^2r+3b^2r-4ar+r-2a^2-2b^2+2a>0
\end{aligned}$$

xaxis: The x-axis ellipse problem [10]

$$\begin{aligned}
&\forall x\forall y\ (0<a\le 1\ \wedge\ 0<b\le 1\ \wedge\\
&0\le c<1-a\ \wedge(c-a<x<c+a\ \wedge\\
&(b^2(x-c)^2+a^2y^2-a^2b^2=0)\ \Rightarrow\ x^2+y^2\ge 1))
\end{aligned}$$

portfolio [71]

$$\begin{array}{ll} \text{Minimize} & 45x_3^2 - 30x_2x_3 + 10x_1x_3 + 3x_2^2 - 40x_1x_2 + 8x_1^2 \\ \text{subject to} & x_1 + x_2 + x_3 \le 10000, \\ & 5x_1 - 4x_2 + 15x_3 \ge 100000, \\ & x_1 \ge 0, x_2 \ge 0, x_3 \ge 0. \end{array}$$

The associated QE problem is given as:

$$\begin{array}{l} \exists x_1 \exists x_2 \exists x_3 \quad (x_1 \ge 0 \ \wedge \ x_2 \ge 0 \ \wedge \ x_3 \ge 0 \ \wedge \\ \quad x_1 + x_2 + x_3 \le 10000 \ \wedge \ 5x_1 - 4x_2 + 15x_3 \ge 100000 \ \wedge \\ \quad y = 45x_3^2 - 30x_2x_3 + 10x_1x_3 + 3x_2^2 - 40x_1x_2 + 8x_1^2). \end{array}$$

port-nox3 portfolio problem removed redundant constraint [71]:

$$\begin{array}{ll} \text{Minimize} & 45x_3^2 - 30x_2x_3 + 10x_1x_3 + 3x_2^2 - 40x_1x_2 + 8x_1^2 \\ \text{subject to} & x_1 + x_2 + x_3 \le 10000, \\ & 5x_1 - 4x_2 + 15x_3 \ge 100000, \\ & x_1 \ge 0, x_2 \ge 0. \end{array}$$

The associated QE problem is given as:

$$\begin{array}{l} \exists x_1 \exists x_2 \exists x_3 \quad (x_1 \ge 0 \ \wedge \ x_2 \ge 0 \ \wedge \\ \quad x_1 + x_2 + x_3 \le 10000 \ \wedge \ 5x_1 - 4x_2 + 15x_3 \ge 100000 \ \wedge \\ \quad y = 45x_3^2 - 30x_2x_3 + 10x_1x_3 + 3x_2^2 - 40x_1x_2 + 8x_1^2). \end{array}$$

port-para parametric optimization problem of [71]:

$$\begin{array}{ll} \text{Minimize} & 45x_3^2 - 30x_2x_3 + 10x_1x_3 + 3x_2^2 - 40x_1x_2 + 8x_1^2 \\ \text{subject to} & x_1 + x_2 + x_3 = t, \\ & 5x_1 - 4x_2 + 15x_3 \ge 100000, \\ & x_1 \ge 0, x_2 \ge 0, x_3 \ge 0. \end{array}$$

The associated QE problem is given as:

$$\begin{array}{l} \exists x_1 \exists x_2 \exists x_3 \quad (x_1 \ge 0 \ \wedge \ x_2 \ge 0 \ \wedge \ x_3 \ge 0 \ \wedge \\ \qquad x_1 + x_2 + x_3 = t \ \wedge \ 5x_1 - 4x_2 + 15x_3 \ge 100000 \ \wedge \\ \qquad y = 45x_3^2 - 30x_2x_3 + 10x_1x_3 + 3x_2^2 - 40x_1x_2 + 8x_1^2). \end{array}$$

Eliminating $x_3$ using *virtual substitution* [55] we obtain the following QE problem:

$$\begin{array}{l} \exists x_1 \exists x_2 \quad (x_1 \ge 0 \ \wedge \ x_2 \ge 0 \ \wedge \\ \qquad t \ge x_1 + x_2 \ \wedge \ 15t - 10x_1 - 19x_2 \ge 100000 \ \wedge \\ \qquad y = 45t^2 + 80tx_1 + 120tx_2 - 43x_1^2 - 70x_1x_2 - 78x_2^2). \end{array}$$

kinoshita

$$\begin{array}{ll} \text{Minimize} & a_1 + a_2 + a_3 \\ \text{subject to} & 1 + C_2^2 + C_3^2 - 100a_i \leq 0 \ (i = 1, \ldots, 3), \\ & a_i \geq 0 \ (i = 1, \ldots, 3), \\ & C_2 = a_1a_3 + a_1 + a_3 + 1, \\ & C_3 = a_1a_2 + a_1 + a_2 + 1. \end{array}$$

The associated QE problem is given as:

$$\begin{array}{l} \exists a_1 \exists a_2 \exists a_3 (t = a_1 + a_2 + a_3 \wedge \\ \quad 1 + (a_1a_3 + a_1 + a_3 + 1)^2 + (a_1a_2 + a_1 + a_2 + 1)^2 \leq 100a_1 \wedge \\ \quad 1 + (a_1a_3 + a_1 + a_3 + 1)^2 + (a_1a_2 + a_1 + a_2 + 1)^2 \leq 100a_2 \wedge \\ \quad 1 + (a_1a_3 + a_1 + a_3 + 1)^2 + (a_1a_2 + a_1 + a_2 + 1)^2 \leq 100a_3 \wedge \\ \quad a_1 \geq 0 \wedge a_2 \geq 0 \wedge a_3 \geq 0). \end{array}$$

mooea extended problem of example 1 in [25, p. 11]:

$$\begin{array}{ll} \text{Minimize} & x_1^2 + x_2^2 + x_3 \text{ and} \\ \text{minimize} & (x_1 - 1)^2 + x_2^2 + x_3 \\ \text{subject to} & -2 \leq x_1 \leq 2, -2 \leq x_2 \leq 2, -1 \leq 10x_3 \leq 1. \end{array}$$

The associated QE problem is given as:

$$\begin{array}{l} \exists x_1 \exists x_2 \exists x_3 \quad (y_1 = x_1^2 + x_2^2 + x_3 \wedge \\ \quad y_2 = (x_1 - 1)^2 + x_2^2 + x_3 \wedge \\ \quad -2 \leq x_1 \leq 2 \ \wedge \ -2 \leq x_2 \leq 2 \ \wedge \ -1 \leq 10x_3 \leq 1). \end{array}$$

wilson [84]

$$\begin{array}{ll} \text{Minimize} & (x_1 - 2)^2 + (x_2 - 1)^2 \text{ and} \\ \text{minimize} & x_1^2 + (x_2 - 6)^2 \\ \text{subject to} & 2/5 \leq x_1 \leq 8/5, 2 \leq x_2 \leq 5. \end{array}$$

The associated QE problem is given as:

$$\begin{array}{l} \exists x_1 \exists x_2 \quad ( y_1 = (x_1 - 2)^2 + (x_2 - 1)^2 \ \wedge \ y_2 = x_1^2 + (x_2 - 6)^2 \ \wedge \\ \quad 2/5 \leq x_1 \leq 8/5 \ \wedge \ 2 \leq x_2 \leq 5). \end{array}$$

lampinen [53, p. 6]

$$\begin{array}{ll} \text{Minimize} & x_1^2+x_2 \text{ and} \\ \text{minimize} & x_1+x_2^2 \\ \text{subject to} & -10 \le x_1 \le 10, -10 \le x_2 \le 10. \end{array}$$

The associated QE problem is given as:

$$\exists x_1 \exists x_2$$
$$(y_1 = x_1^2 + x_2 \;\wedge\; y_2 = x_1 + x_2^2 \;\wedge -10 \le x_1 \le 10 \;\wedge\; -10 \le x_2 \le 10).$$

kimura5

$$\begin{aligned} &\exists x_1 \exists x_2 \exists y_1 \exists y_2 \; ( \; (l_1 l_2 + l_1 + l_2) x_1 x_2 = l_1^2 l_2 + l_1 l_2^2 + l_1 l_2 \;\wedge \\ &\quad (l_1 l_2 + l_1 + l_2)(x_1 + x_2) = l_1^2 + l_2^2 + l_1^2 l_2 + l_1 l_2^2 + l_1 + l_2 \;\wedge \\ &\quad (l_1^2 l_2^2 + l_1^2 + l_2^2) y_1 y_2 = l_1^4 l_2^2 + l_1^2 l_2^4 + l_1^2 l_2^2 \;\wedge \\ &\quad (l_1^2 l_2^2 + l_1^2 + l_2^2)(y_1 + y_2) = l_1^4 + l_2^4 + l_1^4 l_2^2 + l_1^2 l_2^4 + l_1^2 + l_2^2 \;\wedge \\ &\quad l_1 \ge l_2 \ge 1 \;\wedge\; y_1 > x_1 \ge x_2 \;\wedge\; y_1^2 \ge y_2^2 \;\wedge\; y_1 \ge 0). \end{aligned}$$

kimurac

$$\begin{aligned} &\exists x_1 \exists x_2 \exists y_1 \exists y_2 ( \\ &\quad 3(x_1 + x_2) = 2(l_1 + l_2 + 1) \;\wedge \\ &\quad 3 x_1 x_2 = -l_1^2 - l_2^2 + 2 l_1 l_2 + 2 l_1 + 2 l_2 - 1 \;\wedge \\ &\quad (l_1^2 + l_2^2 + l_1^2 l_2^2)(y_1^2 + y_2^2) = l_1^4 + l_2^4 + l_1^4 l_2^2 + l_1^2 l_2^4 + l_1^2 + l_2^2 \;\wedge \\ &\quad (l_1^2 + l_2^2 + l_1^2 l_2^2) y_1^2 y_2^2 = l_1^2 l_2^2 + l_1^4 l_2^2 + l_1^2 l_2^4 \;\wedge \\ &\quad x_1 \ge x_2 \wedge y_1^2 \ge y_2^2 \wedge y_1 \ge 0 \wedge y_1 = x_1). \end{aligned}$$

# Chapter 3

# A Special QE algorithm for Sign Definite Conditions

## 3.1 Introduction

In this chapter we focus on one particular input formula,

$$\forall x \ (x \geq 0 \to f(x) > 0) \tag{3.1}$$

where $f(x)$ is a univariate polynomial with real parameters, which we call a *sign definite condition (SDC)*. The importance of this formula is that many practical engineering problems such as control system design problems can be recast as SDCs [3]. We note that we mainly consider the case where the coefficients of $f$ contain some parameters. An effective QE algorithm for SDCs was proposed in [45] based on a combinatorial approach using a real root counting technique. The Sturm-Habicht sequence [36] is used as the real root counting method.

To improve the efficiency of the proposed method in [45], *simplification* of output logical formulae is a critical issue. Our focus is on developing an effective algorithm which produces formulae that are as simple as possible.

For this purpose, we propose two approaches. First, we use a necessary condition for the SDC to simplify an output formula algebraically. The necessary condition enables us to eliminate extraneous sign combinations derived from real root counting using the Sturm-Habicht sequence. Second, we use Boolean function manipulation. We obtain simple formulae by using the idea of *don't cares* for handling sign conditions that no real numbers satisfy. A *don't care* is an input where a function is not specified. These improvements significantly simplify the output formula of a specialized QE algorithm for the SDC. We also present experimental results for demonstrating the effectiveness of our proposed method.

The organization of this chapter is as follows. Section 3.2 explains the Sturm-Habicht sequence and also an outline of the specialized QE algorithm for the SDC using the Sturm-Habicht sequence. In Section 3.3, we show a necessary condition for the SDC. Simplification of Boolean expressions based on Boolean function manipulation is shown in Section 3.4. The proposed improvements for speeding up the specialized QE algorithm for the SDC are discussed based on experimental results in Section 3.5. Concluding remarks are made in Section 3.6.

## 3.2 Quantifier Elimination for Sign Definite Condition

In this section, we define an SDC and a specialized QE algorithm based on the Sturm-Habicht sequence. Let us denote the fields of real numbers by $\mathbb{R}$.

### 3.2.1 Sign Definite Conditions and Real Root Counting by Sturm-Habicht Sequences

We first introduce a sign definite condition.

**Definition 28.** *Let $f(x)$ be a polynomial in $x$ over $\mathbb{R}$. We call the following condition a* sign definite condition *(SDC) for $f(x)$:*

$$\forall x\ (x \geq 0 \to f(x) > 0).$$

*In addition, we call this the $n$-*th SDC problem *if the leading coefficient of $f$ is not equal to zero and the degree of $f$ is equal to $n$.*

Many important design specifications frequently used as indices of robustness, such as $H_\infty$ norm constraints and stability margins, reduce to SDCs [3]. A typical example is the (frequency restricted) $H_\infty$ norm constraint. An $H_\infty$ norm constraint of a strictly proper transfer function $P(s) = n(s)/d(s)$ expressed as

$$\|P(s)\|_\infty := \sup_\omega |P(j\omega)| < \gamma$$

is equivalent to $\forall \omega\ (\gamma^2 d(j\omega)d(-j\omega) > n(j\omega)n(-j\omega))$, where $j$ denotes the imaginary unit. Since we can find a function $f(\omega^2)$ which satisfies $f(\omega^2) = \gamma^2 d(j\omega)d(-j\omega) - n(j\omega)n(-j\omega) > 0$, letting $x = \omega^2$ leads to an SDC. Similarly, a finite frequency $H_\infty$ norm defined by

$$\|P(s)\|_{[\omega_1,\omega_2]} := \sup_{\omega_1 \leq \omega \leq \omega_2} |P(j\omega)| < \gamma$$

can be recast as the condition $f(x) \neq 0$ in $[\ -\omega_2^2, -\omega_1^2\ ]$, which may be reduced to an SDC for $f(z)$ by the bilinear transformation $z = -(x+\omega_2^2)/(x+\omega_1^2)$. A specialized QE algorithm for SDCs was proposed in [45]. This algorithm uses the following proposition and realizes QE by real root counting using the Sturm-Habicht sequence [36].

**Proposition 29.** *The SDC for a polynomial in $\mathbb{R}[x]$ with a positive leading coefficient is equivalent to the condition that the polynomial has no real root in $x \geq 0$.*

We next describe the Sturm-Habicht sequence.

**Definition 30.** *Let $f(x)$ be a polynomial in $\mathbb{R}[x]$ with degree $n$. The Sturm-Habicht sequence associated to $f$ is defined as the sequence of polynomials $\mathsf{SH}(f) = \{\mathsf{SH}_j(f)\}_{j=0,\dots,n}$ such that $\mathsf{SH}_n(f) = f$, $\mathsf{SH}_{n-1}(f) = \frac{df}{dx}$, $\mathsf{SH}_j(f) = \delta_{n-j}\ \mathrm{Sres}_j(f, \frac{df}{dx})$ for $j \in \{0, \dots, n-2\}$, where $\delta_k = (-1)^{\frac{k(k-1)}{2}}$ and $\mathrm{Sres}_k(f,g)$ is the $k$-th subresultant which is defined as the determinant of the $k$-th Sylvester matrix of $f$ and $g$.*

**Definition 31.** *We define the* sign *of a real number as $1$, $0$, or $-1$ if the number is positive, zero, or negative, respectively. Let $A = \{a_m, \dots, a_0\}$ be a finite sequence of real numbers. We define the* number of sign variations *$V(A)$ by the following rules:*

- *we count 1 sign variation for the groups: $\{-1,+1\}$, $\{+1,-1\}$, $\{-1,0,+1\}$, $\{+1,0,-1\}$, $\{-1,0,0,+1\}$, $\{+1,0,0,-1\}$,*
- *we count 2 sign variations for the groups: $\{+1,0,0,+1\}$, $\{-1,0,0,-1\}$.*

*Let $S(x) = \{S_n(x), S_{n-1}(x), \dots, S_0(x)\}$ be a finite sequence of polynomials in $\mathbb{R}[x]$ and let $\alpha$ be a real number. We construct a sequence $\{h_s, \dots, h_0\}$ of polynomials in $\mathbb{R}[x]$ obtained from $S(x)$ by deleting polynomials identical to zero. The number of sign variations $V_\alpha(S)$ is defined by $V(\{h_s(\alpha), \dots, h_0(\alpha)\})$.*

We note that the sign sequences $\{+1,0,+1\}$, $\{-1,0,-1\}$ and $\{0,0,0\}$ cannot appear by [36].

**Theorem 32.** *(real root counting by the Sturm-Habicht sequence [36]) Let $f$ be a polynomial in $\mathbb{R}[x]$ and let $\alpha$, $\beta$ in $\mathbb{R} \cup \{-\infty, +\infty\}$ with $\alpha < \beta$ and $f(\alpha)f(\beta) \neq 0$. Then $V_\alpha(\mathsf{SH}(f)) - V_\beta(\mathsf{SH}(f))$ is equal to the number of real roots of $f(x)$ in the interval $[\alpha, \beta]$.*

Using Theorem 32, we can obtain the number of real roots of a polynomial in the interval $[0, +\infty)$.

In the rest of this chapter, we denote the sign of $\mathsf{SH}_k(f)$ at $x = \infty$ and $x = 0$ by $s_k$ and $c_k$, respectively.

**Remark 33.** *Let $\mathsf{SH}_k(f) = a_{k,k}x^k + a_{k,k-1}x^{k-1} + \cdots + a_{k,0}$. Then $s_k = 0$ is equivalent to $a_{k,i} = a_{k,k-1} = \cdots = a_{k,0} = 0$ and $s_k > 0$ is equivalent to $(a_{k,k} > 0) \vee (a_{k,k} = 0 \wedge a_{k,k-1} > 0) \vee \cdots \vee (a_{k,k} = a_{k,k-1} = \cdots = a_{k,1} = 0 \wedge a_{k,0} > 0)$. That is, $s_k = 0$ if and only if $\mathsf{SH}_k(f)$ is identically zero. We note that $c_k$ is equivalent to the sign of $a_{k,0}$.*

**Example 34.** *Let $f(x) = 25x^5 + 25x^4 + 10x^3 + 2x^2 + 25x + 1$. The Sturm-Habicht sequence associated to $f$ is as follows:*

$$\begin{array}{rclcl}
\mathsf{SH}_5(f) & = & f(x) & = & 25x^5 + 25x^4 + 10x^3 + 2x^2 + 25x + 1, \\
\mathsf{SH}_4(f) & = & \frac{df}{dx}(x) & = & 125x^4 + 100x^3 + 30x^2 + 4x + 25, \\
\mathsf{SH}_3(f) & = & \delta_{5-3}\mathrm{Sres}_3(f, \frac{df}{dx}) & = & -(310000x), \\
\mathsf{SH}_2(f) & = & \delta_{5-2}\mathrm{Sres}_2(f, \frac{df}{dx}) & = & -(0), \\
\mathsf{SH}_1(f) & = & \delta_{5-1}\mathrm{Sres}_1(f, \frac{df}{dx}) & = & +(1906624000000x), \\
\mathsf{SH}_0(f) & = & \delta_{5-0}\mathrm{Sres}_0(f, \frac{df}{dx}) & = & +(945685504000000).
\end{array}$$

*Thus*

$$\begin{array}{rcl}
\{s_5, s_4, s_3, s_2, s_1, s_0\} & = & \{+1, +1, -1, 0, +1, +1\}, \\
\{c_5, c_4, c_3, c_2, c_1, c_0\} & = & \{+1, +1, 0, 0, 0, +1\}.
\end{array}$$

*Therefore $f(x)$ has no real root for $x \geq 0$ because $V_0(\mathsf{SH}(f)) = V_\infty(\mathsf{SH}(f)) = 2$.*

**Remark 35.** *We note that $s_n = s_{n-1}$ and $s_0 = c_0$ for a polynomial with degree $n$.*

**Definition 36.** $\mathsf{SH}_k(f)$ *is* regular *when the degree of $\mathsf{SH}_k(f)$ is equal to $k$.*

**Theorem 37.** *(Sturm-Habicht Structure Theorem [36]) Let $f$ be a polynomial in $\mathbb{R}[x]$ with degree $n$. Then for every $k \in \{1, \ldots, n-1\}$ such that $\mathsf{SH}_{k+1}(f)$ is regular and* $\deg(\mathsf{SH}_k(f)) = r \leq k$, *we have:*

*(A) if $r < k - 1$, then $\mathsf{SH}_{k-1}(f) = \cdots = \mathsf{SH}_{r+1}(f) = 0$,*

*(B) if* $r < k$*, then* $\mathrm{lc}(\mathsf{SH}_{k+1}(f))^{k-r}\mathsf{SH}_r(f) = \delta_{k-r}\mathrm{lc}(\mathsf{SH}_k(f))^{k-r}\mathsf{SH}_k(f)$,

*(C)* $\mathrm{lc}(\mathsf{SH}_{k+1}(f))^{k-r+2}\mathsf{SH}_{r-1}(f) = \delta_{k-r+2}\mathrm{Prem}(\mathsf{SH}_{k+1}(f), \mathsf{SH}_k(f))$,

*where* $\mathrm{lc}(g)$ *is the leading coefficient of the polynomial* $g$*, and* $\mathrm{Prem}(g, h)$ *is a pseudo remainder of the polynomial* $g$ *by the polynomial* $h$ *defined by*

$$\mathrm{Prem}(g, h) = \mathrm{remainder}(\mathrm{lc}(h)^{\deg(g)-\deg(h)+1}g, h).$$

### 3.2.2 A specialized QE algorithm for SDC

In this subsection, we describe an implementation for a specialized QE algorithm for SDCs [45]. The flow of the algorithm for the $n$-th SDC problem is as follows:

1. consider all the $3^{2n+1}$ (at most) possible sign conditions over $s_k$ and $c_k$,
2. choose all sign conditions $\varphi_n$ which satisfy $V_0(\mathsf{SH}(f)) - V_\infty(\mathsf{SH}(f)) = 0$,
3. compute the Sturm-Habicht sequence associated to $f$,
4. construct semi-algebraic sets generated by coefficients of polynomials in $\mathsf{SH}(f)$ for each selected sign condition and take their union.

Since steps 1 and 2 are independent of the input polynomial, we can execute these steps beforehand and store the results in a database. This greatly improves the total efficiency of the algorithm. In fact, a QE computation for the fifth SDC problem $\forall x(x \geq 0 \rightarrow x^5 + \sum_{i=0}^{4} a_i x^i > 0)$ using the cylindrical algebraic decomposition (CAD) algorithm [20], which is a general QE algorithm and a real root classification algorithm [89] implemented as a Maple command, did not terminate after an hour. In contrast, the specialized algorithm took less than a second.

The result obtained from the above procedure obviously tends to be large and complicated, and hence we should reduce the admissible sign conditions $\varphi_n$ as much as possible. The simplification of $\varphi_n$ makes the algorithm and post-processing, for example drawing the feasible regions, more efficient. For example, formulae can be simplified by using these well-known rules:

$$\begin{cases} < \cup > & \leftrightarrow \quad \neq, \\ < \cup = & \leftrightarrow \quad \leq, \\ > \cup = & \leftrightarrow \quad \geq . \end{cases} \tag{3.2}$$

The goal of this chapter is to obtain an output formula equivalent to the SDC by simplifying the possible sign conditions $\varphi_n$.

Table 3.1: $\varphi_2$: sign conditions for the 2nd SDC problem

| $s_2$ | $s_1$ | $s_0$ | $c_2$ | $c_1$ | $c_0$ | $V_0(\mathsf{SH})$ |
|---|---|---|---|---|---|---|
| $+$ | $+$ | $-$ | $+$ | $-$ | $-$ | 1 |
| $+$ | $+$ | $-$ | $+$ | 0 | $-$ | 1 |
| $+$ | $+$ | $-$ | $+$ | $+$ | $-$ | 1 |
| $+$ | $+$ | 0 | $+$ | 0 | 0 | 0 |
| $+$ | $+$ | 0 | $+$ | $+$ | 0 | 0 |
| $+$ | $+$ | $+$ | $+$ | $+$ | $+$ | 0 |

**Example 38.** *Let $f$ be a quadratic polynomial in $\mathbb{R}[x]$. We consider the second SDC problem $\forall x\ (x \geq 0 \rightarrow f(x) > 0)$. Table 3.1 shows sign conditions which satisfy $V_0(\mathsf{SH}(f)) - V_\infty(\mathsf{SH}(f)) = 0$. Each row shows the signs of $s_k$ and $c_k$ when $f$ has no real root in $x \geq 0$. We note that $s_2 = s_1 > 0$ and $s_0 = c_0$ from Remark 35, and that $c_2 > 0$ because $f(0) > 0$ implies $c > 0$. From Table 3.1, we obtain the following quantifier-free formula:*

$$\begin{array}{ll}(s_0 < 0 \wedge c_2 > 0 \wedge c_1 < 0) & \vee \\ (s_0 < 0 \wedge c_2 > 0 \wedge c_1 = 0) & \vee \\ (s_0 < 0 \wedge c_2 > 0 \wedge c_1 > 0) & \vee \\ (s_0 = 0 \wedge c_2 > 0 \wedge c_1 = 0) & \vee \\ (s_0 = 0 \wedge c_2 > 0 \wedge c_1 > 0) & \vee \\ (s_0 > 0 \wedge c_2 > 0 \wedge c_1 > 0). & \end{array} \tag{3.3}$$

*The formula* (3.3) *can be simplified as follows by using* (3.2)*:*

$$\begin{array}{ll}(s_0 < 0 \wedge c_2 > 0) & \vee \\ (s_0 = 0 \wedge c_2 > 0 \wedge c_1 \geq 0) & \vee \\ (s_0 > 0 \wedge c_2 > 0 \wedge c_1 > 0). & \end{array} \tag{3.4}$$

*By constructing formula* (3.4) *beforehand, the QE computation is done by computing the Sturm-Habicht sequence and substitution for $s_k$ and $c_k$. Moreover, simplification of $\varphi_2$ reduces the number of substitutions.*

## 3.3 Necessary Condition for SDC

Now we again consider the formula (3.4) in Example 38. For any quadratic polynomial $f \in R[x]$, $s_0 = 0 \wedge c_2 > 0 \wedge c_1 = 0$ does not hold. In fact, the Sturm-Habicht sequence associated to $f(x) = x^2 + p_1x + p_0$ is $\mathsf{SH}(f) = \{x^2 + p_1x + p_0, 2x + p_1, p_1^2 - 4p_0\}$. In this case, if $s_0 = p_1^2 - 4p_0 = 0$ and $c_1 = p_1 = 0$, we have $p_1 = p_0 = 0$. Thus, we obtain $c_2 = p_1 = 0$. From this fact, we can reduce formula (3.4) to the following:

$$\begin{array}{ll} (s_0 < 0 \wedge c_2 > 0) & \vee \\ (s_0 \geq 0 \wedge c_2 > 0 \wedge c_1 > 0). & \end{array}$$

In this section, to simplify the formula derived from the possible sign conditions $\varphi_n$, we give a necessary condition for a sign sequence of the Sturm-Habicht sequence associated to a polynomial which satisfies the SDC. We use the notation introduced in the previous section.

The following theorem provides a necessary condition.

**Theorem 39.** *Let $f = \sum_{i=0}^n p_i x^i$ be a polynomial in $\mathbb{R}[x]$ where $p_n \neq 0$, and let $u$ be the smallest nonnegative integer $k$ such that $s_k \neq 0$. We define $s_k$ and $c_k$ to be zero when $k < 0$ or $k > n$. When $f$ satisfies $p_0 > 0$ and $p_n > 0$, the following conditions hold:*

$$\begin{aligned}
& s_n > 0, s_{n-1} > 0, c_n > 0, s_0 = c_0, \\
& s_k = 0 \to c_k = 0, \ (\forall k \in \{0, \dots, n-2\}), \\
& c_u \neq 0, \\
& c_{n-1} = 0 \to c_{n-2} < 0, \\
& s_{n-2} = 0 \to s_{n-3} = \cdots = s_0 = 0, \\
& c_{k+2} \neq 0 \wedge c_{k+1} = 0 \to c_k \neq c_{k+2}, \ (\forall k \in \mathcal{N} = \{u, \dots, n-2\}), \\
& c_k = c_{k+1} = 0 \wedge c_{k-1}c_{k+2}s_ks_{k+1} \neq 0 \to s_ks_{k+2} < 0, \ (\forall k \in \mathcal{N}), \\
& c_k = \cdots = c_{k+m} = 0 \to s_{k+1} = \cdots = s_{k+m-1} = 0, \ (\forall k \in \mathcal{N}, m > 1), \\
& s_{k+2} = 0 \wedge s_{k+1} \neq 0 \to s_k \neq 0, \ (\forall k \in \mathcal{N}), \\
& s_{k-1} \neq 0 \wedge s_k = \cdots = s_{k+m} = 0 \wedge s_{k+m+1} \neq 0 \to s_{k+m+2}^m s_{k-1} = \delta_{m+2} s_{k+m+1}^{m+1} \\
& \qquad \wedge s_{k+m+2}^m c_{k-1} = \delta_{m+2} s_{k+m+1}^m c_{k+m+1}, \ (\forall k \in \mathcal{N}, m \geq 0).
\end{aligned}$$

The theorem was proved by proving the following ten lemmas. Lemma 42 and part of Lemma 45 are already mentioned in [36]. We note that $p_0 > 0$ because $f(0) = p_0 > 0$, and $p_n > 0$ so that $f(x) > 0$ is satisfied for sufficiently large $x > 0$.

The following lemma is obvious.

**Lemma 40.** *Let $f = \sum_{i=0}^n p_i x^i$ be a polynomial in $\mathbb{R}[x]$. When $p_0 > 0$ and $p_n > 0$, $s_n = s_{n-1} = c_n > 0$ and $s_0 = c_0$ hold.*

For the rest of this section, we assume that the leading coefficient $p_n$ and the constant term $p_0$ of a given polynomial $f$ are positive. For the sake of simplicity, we denote $\mathsf{SH}_k(f)$ by $\mathsf{SH}_k$.

**Lemma 41.** *For $k \in \{0, \dots, n\}$, $s_k = 0$ implies that $c_k = 0$.*

*Proof.* From Remark 33, $s_k = 0$ implies that $\mathsf{SH}_k = 0$. Then $c_k = 0$. □

**Lemma 42.** *$c_u \neq 0$, where $u$ is the smallest nonnegative integer $k$ such that $s_k \neq 0$.*

*Proof.* This lemma is mentioned in [36]. From the definition of $u$, $\mathsf{SH}_u$ is the greatest common factor of $f$ and $df/dx$. Because of the assumption $f(0) \neq 0$, $\mathsf{SH}_u(0) \neq 0$. Thus $c_u \neq 0$. □

**Lemma 43.** *$c_{n-1} = 0$ implies $c_{n-2} < 0$.*

*Proof.* By definition, we have $\mathsf{SH}_{n-1}(0) = p_1$ and $\mathsf{SH}_{n-2}(0) = p_1 p_{n-1} p_n - n^2 p_0 p_n^2$. Since $p_n$ and $p_0$ are positive by the assumption, and $c_{n-1}$ is the sign of $p_1$, $\mathsf{SH}_{n-2}(0) = -n^2 p_0 p_n^2 < 0$ . □

**Lemma 44.** *$s_{n-2} = 0$ implies $s_{n-3} = \cdots = s_1 = s_0 = 0$.*

*Proof.* Suppose that there exists some integer $k \leq n-3$ such that $s_k \neq 0$. From Theorem 37, the degree of $\mathsf{SH}_{n-1}$ must be less than $n-2$. Since the leading coefficient of $f$ is positive and $\mathsf{SH}_{n-1} = df/dx$, this is a contradiction. □

**Lemma 45.** *For $k \in \{1, \dots, n-1\}$, $c_{k-1} \neq c_{k+1}$ if $c_{k+1} \neq 0$ and $c_k = 0$.*

*Proof.* We only need to consider five cases. Case (1) is proved in [36].

(1) $\deg(\mathsf{SH}_k) = r, \deg(\mathsf{SH}_{k+1}) = k+1, \mathsf{SH}_k(0) = c_k = 0$:

   (a) $r = k$: From Theorem 37 (C), $\mathrm{lc}(\mathsf{SH}_{k+1})^2 \mathsf{SH}_{k-1} = -\mathrm{Prem}(\mathsf{SH}_{k+1}, \mathsf{SH}_k)$. Then $c_{k-1} = -c_{k+1}$. Since $f(0) \neq 0$, $x$ is not a common divisor of $\mathsf{SH}_k$ and $\mathsf{SH}_{k+1}$. Then $c_{k+1} \neq 0$. Hence we obtain $c_{k+1} \neq c_{k-1}$.

   (b) $r < k$: From Theorem 37 (A) and (C), $c_{k-1} = 0$.

(2) $\deg(\mathsf{SH}_{k+1}) = r < k+1, \deg(\mathsf{SH}_{k+2}) = k+2, \mathsf{SH}_k(0) = c_k = 0$:

   (a) $r < k-1$: From Theorem 37 (A), $c_{k-1} = 0$.

   (b) $r = k-1$: From Theorem 37 (B), $\mathrm{lc}(\mathsf{SH}_{k+2})^2 \mathsf{SH}_{k-1} = -\mathrm{lc}(\mathsf{SH}_{k+1})^2 \mathsf{SH}_{k+1}$. Then $\mathsf{SH}_{k-1}\mathsf{SH}_{k+1} \leq 0$.

(c) $r = k$: From Theorem 37 (B), $c_{k+1} = 0$.

□

**Lemma 46.** *For $k \in \{u+1, \ldots, n-2\}$, $c_{k-1} \neq 0 \wedge c_k = c_{k+1} = 0 \wedge c_{k+2} \neq 0 \wedge s_k \neq 0 \wedge s_{k+1} \neq 0 \to s_{k+2}s_k < 0$.*

*Proof.* From Lemma 41, $\mathsf{SH}_{k+2} \neq 0$ because $c_{k+2} \neq 0$. We first consider the case where $\mathsf{SH}_{k+2}$ is not regular. From Theorem 37 (A), $\mathsf{SH}_{k+1}$ is regular. In addition, from Theorem 37 (B), $\mathsf{SH}_{k+1}$ is a constant multiple of $\mathsf{SH}_{k+2}$. Thus when $\mathsf{SH}_{k+2}$ is not regular, $c_{k+2} \neq 0$ and $c_{k+1} = 0$ are not satisfied simultaneously.

We next consider the case where $\mathsf{SH}_{k+2}$ is regular. We assume that $\mathsf{SH}_{k+1}$ is regular. From Theorem 37 (C), $\mathrm{lc}(\mathsf{SH}_{k+2})^2\mathsf{SH}_k = \delta_2\mathrm{Prem}(\mathsf{SH}_{k+2}, \mathsf{SH}_{k+1})$ . Since $c_k = c_{k+1} = 0$, $x$ must be a common divisor of $\mathsf{SH}_k$ and $\mathsf{SH}_{k+1}$. This is a contradiction to $f(0) \neq 0$. Therefore $\mathsf{SH}_{k+1}$ is not regular and the degree of $\mathsf{SH}_{k+1}$ is $k$ from Theorem 37 (A). From Theorem 37 (B), we obtain $s_{k+2}s_k < 0$. □

**Lemma 47.** *For $k \in \{u+1, \ldots, n-2\}$ and $m \in \{2, \ldots, n-k-1\}$, $c_{k+m+1} \neq 0 \wedge c_{k+m} = \cdots = c_k = 0 \to s_{k+m-1} = \cdots = s_{k+1} = 0$ .*

*Proof.* (i) Case $s_{k+m} \neq 0$. By the proof of Lemma 46, $\mathsf{SH}_{k+m+1}$ is regular and $\mathsf{SH}_{k+m}$ is not. We denote the degree of $\mathsf{SH}_{k+m}$ by $d_1$. When $d_1 > k$, by Theorem 37 (C), $\mathsf{SH}_{d_1-1}$ is a constant multiple of a pseudo remainder of $\mathsf{SH}_{k+m+1}$ divided by $\mathsf{SH}_{k+m}$. Since $\mathsf{SH}_{d_1-1}(0) = \mathsf{SH}_{k+m}(0) = 0$, $x$ is a common divisor of $\mathsf{SH}_{d_1-1}$ and $\mathsf{SH}_{k+m}$. This is a contradiction since $f(0) \neq 0$. Thus $d_1 \leq k$. Therefore by Theorem 37 (A), we have $s_{d_1+1} = \cdots = s_{k+m-1} = 0$.

(ii) Case $s_{k+m} = 0$. We denote the degree of $\mathsf{SH}_{k+m+1}$ by $d_2$. When $d_2 \geq k$, by Theorem 37 (B), $\mathsf{SH}_{d_2}$ is a constant multiple of $\mathsf{SH}_{k+m+1}$. This is a contradiction since $\mathsf{SH}_{d_2-1}(0) = 0$ and $\mathsf{SH}_{k+m+1}(0) \neq 0$. Thus $d_2 < k$. Therefore, by Theorem 37 (A), we have $s_{d_2+1} = \cdots = s_{k+m-1} = 0$.

□

**Lemma 48.** *For $k \in \{u, \ldots, n-3\}$, $s_{k+2} = 0 \wedge s_{k+1} \neq 0$ implies $s_k \neq 0$.*

*Proof.* The polynomial $\mathsf{SH}_{k+1}$ is regular from Theorem 37 (B). From Theorem 37 (C), $\mathsf{SH}_k$ is a pseudo remainder of $\mathsf{SH}_r$ divided by $\mathsf{SH}_{k+1}$ for some $r > k+1$. Since $k \geq u$, we obtain $\mathsf{SH}_k \neq 0$. □

| $x$ | $y$ | $x \cdot y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| $x$ | $y$ | $x + y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| $x$ | $x'$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

Figure 3.1: Boolean operations (AND, OR, and NOT)

**Lemma 49.** *For $k \in \{u+1, \dots, n-3\}$ and $m \in \{0, \dots, n-k-1\}$, $s_{k+m+1} \neq 0 \wedge s_{k+m} = \dots = s_k = 0 \wedge s_{k-1} \neq 0$ implies that $s_{k+m+2}^m s_{k-1} = \delta_{m+2} s_{k+m+1}^{m+1} \;\wedge\; s_{k+m+2}^m c_{k-1} = \delta_{m+2} s_{k+m+1}^m c_{k+m+1}$ .*

*Proof.* This is obtained from Theorem 37 (B). □

## 3.4 Simplification of Boolean Expressions

In this section, we explain an approach to simplifying logical formulae by using a simplification method for Boolean expressions based on Boolean function manipulation [9].

### 3.4.1 Boolean Algebra and Simplification of Boolean Expressions

In this subsection, we define a Boolean algebra and a Boolean function.

**Definition 50.** *A* Boolean algebra *is an algebraic system consisting of the set $B = \{0, 1\}$, two binary operations called AND and OR denoted by the symbols $\cdot$ and $+$ respectively, and a unary operation called NOT denoted by a prime, $'$. The definitions of the AND, OR, and NOT operations are as in Figure 3.1.*

**Definition 51.** *A* Boolean variable *is a two-valued variable which can take either of the two distinct values 0 and 1. A* literal *is a Boolean variable or its complement. A* product term *is a literal or a conjunction of literals where no literal appears more than once. A* sum of products *is a product term or a disjunction of product terms.*

**Definition 52.** *A* Boolean expression *is a combination of a finite number of Boolean variables and Boolean constants by means of the Boolean operations defined in Defi-*

*nition 50. A* (completely specified) Boolean function *with $n$ variables is a mapping $f : B^n \to B$.*

**Definition 53.** *An* incompletely specified Boolean function *of $n$ variables is a Boolean function which is defined over a subset of $B^n$. An input combination for which the function is not specified is called a* don't care*.*

In general, there are a number of Boolean expressions that represent a Boolean function. For example, the Boolean expressions $(x+y)'$ and $x'+y'$ represent the same Boolean function. In this chapter, finding a Boolean expression with a relatively small number of product terms is called *simplification* of Boolean expressions.

**Example 54.** *Consider an incompletely specified Boolean function $f$ whose truth table is defined as follows. Entry $d$ in the column "$f$" means that the corresponding function value is unspecified.*

| $x$ | $y$ | $z$ | $f$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | $d$ |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | $d$ |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

*For incompletely specified Boolean functions, we can expand the notion of simplification of Boolean expressions, because an incompletely specified Boolean function represents a set of completely specified Boolean functions. We can choose a completely specified Boolean function by assigning 0 or 1 to each* don't care *entry $d$. By considering both assignments to* don't cares *and the expressions of these functions, we may obtain better expressions. In the above example, when we assign 1 to every $d$, we obtain the simplified Boolean expression $f = x + y'$.*

In integrated circuit design, simplification of Boolean expressions directly corresponds to minimization of the area of the designed circuit. Hence, many efficient techniques have been proposed, such as the heuristic method called ESPRESSO [9] and several exact methods based on binary decision diagrams (BDDs) [23, 61].

### 3.4.2 Simplification of $\varphi_n$ based on Boolean Expression Minimization

In this subsection, we consider simplification of $\varphi_n$ presented in Subsection 3.2.2 by using simplification techniques for Boolean expressions.

First, since the signs of polynomials handled in this chapter are three-valued, we need two Boolean variables to represent them. In this chapter, we use the variables $x$ and $y$ to represent zero, positive, and negative as $x'y'$, $xy'$, and $x'y$, respectively. We then obtain a simplified representation of $\varphi_n$ by applying simplification techniques for Boolean expressions.

**Example 55.** *Consider formula* (3.3) *again. Let us use* $x_1y_1, x_2y_2, x_3y_3$ *to represent the signs of* $s_0$, $c_2$, $c_1$, *respectively. Then we can represent formula* (3.3) *as the following Boolean expression:*

$$x_1'y_1x_2y_2'x_3'y_3 + x_1'y_1x_2y_2'x_3'y_3' + x_1'y_1x_2y_2'x_3y_3' +$$
$$x_1'y_1'x_2y_2'x_3'y_3' + x_1'y_1'x_2y_2'x_3y_3' + x_1y_1'x_2y_2'x_3y_3'.$$

As mentioned in Subsection 3.4.1, introduction of *don't cares* is likely to simplify Boolean expressions further. For the specialized QE algorithm for the SDC, we can introduce *don't cares* as follows.

1. As mentioned above, we use two Boolean variables $x$ and $y$ to represent the sign of a polynomial. While the sign is three-valued, the two Boolean variables can represent four values. Since we do not use $xy$ here, we can consider $xy$ as a *don't care*.
2. Since we do not need to consider sign conditions which do not satisfy Lemma 41 through Lemma 49, we consider them as *don't cares*.
3. In the same way, since it never happens that $V_0(\mathsf{SH}(f)) < V_\infty(\mathsf{SH}(f))$, we can introduce *don't cares* in this case.

## 3.5 Computational Results

In this section, we present computational results.

Table 3.2 shows the results of simplification by our approach. All the computational experiments were executed on a personal computer with an Intel(R) Core(TM) i7-3540M CPU 3.0 GHz and 2.0 GByte memory. We used the ESPRESSO logic minimizer

[1] to simplify formulae. The first column "deg" gives the degree of an input polynomial. The second column "var" gives the number of Boolean variables which are inputs to the ESPRESSO logic minimizer. When the degree of an input polynomial is $n$, the Sturm-Habicht sequence contains $n+1$ elements. Since $s_n = s_{n-1} = c_n > 0$ and $s_0 = c_0$ by Lemma 40, we only consider the sign sequence for $2(n+1)-4 = 2n-2$ polynomials. Hence the number of Boolean variables is $2(2n-2) = 4n-4$. The column "SyN" gives the number of product terms for the previous SyNRAC implementation in [45]. The column "DC" gives the number of product terms for the implementation with an approximate ESPRESSO method in which we introduce a *don't care* only when a sign condition satisfies condition 1 presented in Subsection 3.4.2 or Lemma 41. Both conditions can be obtained without using Theorem 37. The columns "ESP app" and "ESP ex" give the number of product terms for the implementation presented in this chapter with an approximate ESPRESSO method and with an exact ESPRESSO method, respectively. The seventh column "terms" gives the number of sign conditions which satisfy Theorem 37. The last two columns "time app" and "time ex" give the computing times for the approximate and the exact minimization algorithms, respectively. Computing times are given in seconds. We note that since this simplification step is executed beforehand, a comparison of the computing times is not an essential problem in QE computation.

We see that our improvements greatly reduce the number of product terms by comparing "SyN" and "ESP app", and that *don't cares* are important for simplifying a formula by comparing "DC" and "ESP app". We have not obtained the solutions for the seventh and the eighth SDC problems by an exact ESPRESSO method. However, we see that "ESP app" outputs good approximate solutions up to the sixth SDC problem.

Figure 3.2 presents the input and output files for the ESPRESSO software [1] for the third SDC problem. Rows 5 to 42 of the input file indicate output values for each sign condition by Boolean expressions. Rows 6 to 9 of the output file present the simplified formula by Boolean expressions. Each row shows a truth table row consisting of 8 inputs and 1 output. Each position in the input plane corresponds to an input variable, where "0" indicates that the corresponding input literal appears complemented in the product term, and "1" indicates that the input literal appears uncomplemented in the product term, and "-" indicates that the input literal does not appear in the product term. The numbers 1 and 2 at the end of a row indicates that a sign condition is true and a *don't care*, respectively. The symbol "-" in the input file

Table 3.2: Computational results for the SDC problems

| deg | var | SyN | DC | ESP app | ESP ex | term | time app | time ex |
|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 5 | 2 | 2 | 2 | 5 | 0.01 | 0.01 |
| 3 | 8 | 17 | 7 | 4 | 4 | 21 | 0.01 | 0.01 |
| 4 | 12 | 64 | 24 | 10 | 10 | 99 | 0.01 | 0.04 |
| 5 | 16 | 302 | 85 | 18 | 18 | 480 | 0.05 | 1.12 |
| 6 | 20 | 1229 | 299 | 57 | 57 | 2352 | 0.72 | 61.92 |
| 7 | 24 | 5238 | 1096 | 121 | - | 11656 | 21.40 | >350h |
| 8 | 28 | 20468 | 4037 | 353 | - | 58284 | 757.59 | >350h |

implies the input literal does not appear. The products of the Boolean variables $x_0y_0$, $x_1y_1$, $x_2y_2$ and $x_3y_3$ show the signs of $s_1$, $s_0$, $c_2$ and $c_1$, respectively. For example, row 18 shows that $s_1 < 0 \wedge s_0 = c_0 = 0 \wedge c_2 > 0 \wedge c_1 > 0$ and that this sign condition is a *don't care*, because $V_0(\mathsf{SH}(f)) = 0 < 1 = V_\infty(\mathsf{SH}(f))$ which satisfies condition 3 in Section 3.4.2. Rows 28 to 31 show that $xy$ is a *don't care*. Rows 32 to 33, 34 to 35, 36 to 37, 38 to 39 and 40 to 42 are from Lemmas 44, 41, 45, 43 and 42, respectively.

By our approach, the 22 sign conditions for the third SDC problem were reduced to four conditions and we obtained the following formula:

$$(s_1 < 0 \wedge s_0 > 0) \vee (s_1 < 0 \wedge c_1 < 0) \vee (s_0 < 0 \wedge c_1 < 0) \vee (c_2 \geq 0 \wedge c_1 \geq 0).$$

Let $f(x) = x^3 + ax^2 + bx + c$. The Sturm-Habicht sequence associated to $f$ is $\{\mathsf{SH}_3 = f$, $\mathsf{SH}_2 = df/dx = 3x^2 + 2ax + b$, $\mathsf{SH}_1 = (2a^2 - 6b)x + ab - 9c$, $\mathsf{SH}_0 = -4b^3 + a^2b^2 - 4a^3c + 18bac - 27c^2\}$, so we obtain a simple quantifier-free formula:

$$\begin{aligned} c > 0 \wedge \quad & ((2a^2 - 6b < 0 \vee 2a^2 - 6b = 0 \wedge ab - 9c < 0) \wedge \mathsf{SH}_0 > 0 \vee \\ & (2a^2 - 6b < 0 \vee 2a^2 - 6b = 0 \wedge ab - 9c < 0) \wedge ab - 9c < 0 \vee \\ & \mathsf{SH}_0 < 0 \wedge ab - 9c < 0 \ \vee \ b \geq 0 \wedge ab - 9c \geq 0). \end{aligned} \tag{3.5}$$

The formula we obtained using our approach is significantly simpler than that using [45] which has 17 product terms. However, the formula can be simplified further. For example, there do not exist real numbers $a$, $b$, and $c$ such that

$$c > 0 \wedge ((2a^2 - 6b < 0 \vee 2a^2 - 6b = 0 \wedge ab - 9c < 0) \wedge \mathsf{SH}_0 > 0),$$

which is the first product term in (3.5), and thus, we can reduce the formula more. The necessary condition for the SDC presented in Section 3.3 considers only the case that $s_k$ or $c_k$ is zero. To obtain the simpler formula, using $\varphi_n$ to find necessary and sufficient conditions is a promising direction which will be part of our future work.

**Remark 56.** *CAD is an algorithm that constructs simple quantifier-free formulae because CAD uses sign information of many projection factors. In fact, in the third SDC problem we obtain the following simpler formula by CAD [10, 13, 50] using six projection factors:*

$$c > 0 \wedge (b \geq 0 \wedge a \geq 0 \vee \mathsf{SH}_0 < 0).$$

### 3.5.1 Results

In this subsection, we show the solution formulae of SDC problems by our algorithm.

**2nd problem**

$$\begin{array}{lclcl}
p_2 > 0 & \wedge & p_0 > 0 & \wedge & ( \\
 & & c_1 > 0 & \vee & \\
s_0 < 0 & & & & )
\end{array}$$

**3rd problem**

$$\begin{array}{lclclclcl}
p_3 > 0 & \wedge & p_0 > 0 & \wedge & ( & & & & \\
s_1 < 0 & \wedge & s_0 > 0 & & & & & & \vee \\
s_1 < 0 & & & & & \wedge & c_1 < 0 & & \vee \\
 & & s_0 < 0 & & & \wedge & c_1 < 0 & & \vee \\
 & & & & c_2 \geq 0 & \wedge & c_1 \geq 0 & & )
\end{array}$$

**4th problem**

$$
\begin{array}{lllllll}
p_4 > 0 & \wedge\ p_0 > 0 & \wedge\ \ ( & & & & \\
s_2 < 0 & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & & & \vee \\
s_2 < 0 & & & \wedge\ c_3 \geq 0 & & \wedge\ c_1 < 0 & \vee \\
& s_1 < 0 & & \wedge\ c_3 \geq 0 & & \wedge\ c_1 < 0 & \vee \\
& s_1 < 0 & & & \wedge\ c_2 < 0 & \wedge\ c_1 < 0 & \vee \\
s_2 < 0 & & & & \wedge\ c_2 < 0 & \wedge\ c_1 \leq 0 & \vee \\
& & s_0 < 0 & \wedge\ c_3 \geq 0 & & \wedge\ c_1 < 0 & \vee \\
& & s_0 < 0 & & \wedge\ c_2 < 0 & \wedge\ c_1 < 0 & \vee \\
& s_1 < 0 & \wedge\ s_0 > 0 & & & \wedge\ c_1 \geq 0 & \vee \\
s_2 < 0 & \wedge\ s_1 > 0 & & & & \wedge\ c_1 \geq 0 & \vee \\
& & & c_3 \geq 0 & \wedge\ c_2 \geq 0 & \wedge\ c_1 \geq 0 & )
\end{array}
$$

**5th problem**

$$
\begin{array}{lllllllll}
p_5 > 0 & \wedge\ p_0 > 0 & \wedge\ \ ( & & & & & & \\
s_3 < 0 & \wedge\ s_2 > 0 & \wedge\ s_1 < 0 & \wedge\ s_0 > 0 & & & & & \vee \\
& & s_1 < 0 & \wedge\ s_0 > 0 & \wedge\ c_4 \geq 0 & & & \wedge\ c_1 \geq 0 & \vee \\
& & s_1 < 0 & & & \wedge\ c_3 < 0 & \wedge\ c_2 \leq 0 & \wedge\ c_1 < 0 & \vee \\
& & & s_0 < 0 & & \wedge\ c_3 < 0 & \wedge\ c_2 < 0 & \wedge\ c_1 < 0 & \vee \\
& & s_1 < 0 & \wedge\ s_0 > 0 & & & \wedge\ c_2 \geq 0 & \wedge\ c_1 > 0 & \vee \\
s_3 < 0 & & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & & & \wedge\ c_1 \leq 0 & \vee \\
s_3 < 0 & \wedge\ s_2 > 0 & & & \wedge\ c_4 \geq 0 & \wedge\ c_3 \geq 0 & & & \vee \\
& s_2 < 0 & \wedge\ s_1 \geq 0 & & & & \wedge\ c_2 \geq 0 & \wedge\ c_1 > 0 & \vee \\
& s_2 < 0 & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & & & \wedge\ c_1 \leq 0 & \vee \\
& & s_1 < 0 & & \wedge\ c_4 \geq 0 & \wedge\ c_3 \geq 0 & & \wedge\ c_1 < 0 & \vee \\
s_3 < 0 & \wedge\ s_2 \geq 0 & & & & & \wedge\ c_2 > 0 & \wedge\ c_1 \geq 0 & \vee \\
& & s_1 < 0 & \wedge\ s_0 > 0 & & \wedge\ c_3 \leq 0 & & \wedge\ c_1 \geq 0 & \vee \\
s_3 < 0 & \wedge\ s_2 > 0 & \wedge\ s_1 < 0 & & & & & \wedge\ c_1 \leq 0 & \vee \\
& s_2 < 0 & \wedge\ s_1 \geq 0 & & & \wedge\ c_3 < 0 & \wedge\ c_2 \leq 0 & & \vee \\
& & & & c_4 \geq 0 & \wedge\ c_3 > 0 & \wedge\ c_2 \geq 0 & \wedge\ c_1 \geq 0 & \vee \\
& & & s_0 < 0 & \wedge\ c_4 \geq 0 & \wedge\ c_3 \geq 0 & & \wedge\ c_1 \leq 0 & \vee \\
& s_2 \leq 0 & \wedge\ s_1 \geq 0 & & \wedge\ c_4 \geq 0 & \wedge\ c_3 \geq 0 & & & \vee \\
s_3 < 0 & \wedge\ s_2 \geq 0 & & & & \wedge\ c_3 \leq 0 & \wedge\ c_2 \leq 0 & & )
\end{array}
$$

## 6th problem

$$\begin{array}{lllllllllll}
p_6 > 0 & \wedge\ p_0 > 0 & \wedge\ \ ( & & & & & & & & \\
s_4 < 0 & \wedge\ s_3 \geq 0 & \wedge\ s_2 \leq 0 & \wedge\ s_1 > 0 & \wedge\ s_0 < 0 & & & & & & \vee \\
s_4 < 0 & \wedge\ s_3 \geq 0 & \wedge\ s_2 \leq 0 & & & \wedge\ c_5 \geq 0 & & & & \wedge\ c_1 \leq 0 & \vee \\
 & & & & s_0 < 0 & \wedge\ c_5 \geq 0 & & \wedge\ c_3 < 0 & \wedge\ c_2 \leq 0 & \wedge\ c_1 < 0 & \vee \\
s_4 < 0 & & & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & & \wedge\ c_3 \geq 0 & & \wedge\ c_1 < 0 & \vee \\
 & & & & s_0 < 0 & & \wedge\ c_4 \leq 0 & \wedge\ c_3 < 0 & \wedge\ c_2 < 0 & \wedge\ c_1 < 0 & \vee \\
 & s_3 < 0 & & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & & & \wedge\ c_2 \leq 0 & \wedge\ c_1 < 0 & \vee \\
 & s_3 < 0 & & & & \wedge\ c_5 \geq 0 & \wedge\ c_4 > 0 & \wedge\ c_3 \geq 0 & & \wedge\ c_1 \leq 0 & \vee \\
s_4 < 0 & & & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & \wedge\ c_5 \geq 0 & & & & \wedge\ c_1 < 0 & \vee \\
 & & & s_1 < 0 & & & \wedge\ c_4 < 0 & \wedge\ c_3 < 0 & \wedge\ c_2 \leq 0 & \wedge\ c_1 < 0 & \vee \\
 & s_3 < 0 & & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & \wedge\ c_5 \geq 0 & & & & \wedge\ c_1 < 0 & \vee \\
 & s_3 < 0 & \wedge\ s_2 \geq 0 & & & & & \wedge\ c_3 \geq 0 & \wedge\ c_2 > 0 & \wedge\ c_1 \geq 0 & \vee \\
s_4 < 0 & \wedge\ s_3 \geq 0 & \wedge\ s_2 < 0 & \wedge\ s_1 \geq 0 & & & & & \wedge\ c_2 \leq 0 & & \vee \\
 & & s_2 > 0 & & & \wedge\ c_5 \geq 0 & \wedge\ c_4 \geq 0 & \wedge\ c_3 \geq 0 & \wedge\ c_2 > 0 & \wedge\ c_1 \geq 0 & \vee \\
s_4 < 0 & \wedge\ s_3 \geq 0 & \wedge\ s_2 < 0 & & & & \wedge\ c_4 \leq 0 & & & \wedge\ c_1 < 0 & \vee \\
s_4 < 0 & & & \wedge\ s_1 > 0 & & \wedge\ c_5 \geq 0 & & & \wedge\ c_2 > 0 & \wedge\ c_1 \geq 0 & \vee \\
s_4 < 0 & \wedge\ s_3 \geq 0 & & \wedge\ s_1 < 0 & & & & & \wedge\ c_2 \leq 0 & \wedge\ c_1 < 0 & \vee \\
 & & s_2 < 0 & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & \wedge\ c_5 \geq 0 & & & & \wedge\ c_1 < 0 & \vee \\
 & s_3 \leq 0 & \wedge\ s_2 > 0 & \wedge\ s_1 \leq 0 & & \wedge\ c_5 \geq 0 & & & \wedge\ c_2 \geq 0 & & \vee \\
s_4 < 0 & & & \wedge\ s_1 > 0 & \wedge\ s_0 < 0 & & & & \wedge\ c_2 \leq 0 & \wedge\ c_1 < 0 & \vee \\
s_4 < 0 & & & \wedge\ s_1 \geq 0 & & & \wedge\ c_4 < 0 & \wedge\ c_3 < 0 & \wedge\ c_2 \leq 0 & & \vee \\
 & & s_2 < 0 & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & & \wedge\ c_3 \geq 0 & & \wedge\ c_1 \leq 0 & \vee \\
 & s_3 < 0 & & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & & \wedge\ c_3 \geq 0 & & \wedge\ c_1 \leq 0 & \vee \\
s_4 < 0 & & & \wedge\ s_1 > 0 & & \wedge\ c_5 > 0 & & \wedge\ c_3 < 0 & \wedge\ c_2 \leq 0 & & \vee \\
s_4 < 0 & & & \wedge\ s_1 > 0 & \wedge\ s_0 < 0 & & \wedge\ c_4 < 0 & & & \wedge\ c_1 < 0 & \vee \\
 & & & s_1 < 0 & \wedge\ s_0 > 0 & & & \wedge\ c_3 \geq 0 & \wedge\ c_2 > 0 & \wedge\ c_1 \geq 0 & \vee \\
 & s_3 \leq 0 & \wedge\ s_2 > 0 & & & \wedge\ c_5 \geq 0 & & & \wedge\ c_2 > 0 & \wedge\ c_1 \geq 0 & \vee \\
 & & s_2 \geq 0 & & \wedge\ s_0 < 0 & \wedge\ c_5 \geq 0 & \wedge\ c_4 \geq 0 & \wedge\ c_3 \geq 0 & & \wedge\ c_1 \leq 0 & \vee \\
 & & s_2 < 0 & \wedge\ s_1 \geq 0 & & & & \wedge\ c_3 > 0 & \wedge\ c_2 \geq 0 & \wedge\ c_1 > 0 & \vee \\
 & s_3 \leq 0 & \wedge\ s_2 \geq 0 & \wedge\ s_1 < 0 & \wedge\ s_0 > 0 & & & & & \wedge\ c_1 > 0 & \vee \\
 & & s_2 < 0 & & & & \wedge\ c_4 < 0 & \wedge\ c_3 < 0 & \wedge\ c_2 < 0 & \wedge\ c_1 \leq 0 & \vee \\
s_4 < 0 & \wedge\ s_3 > 0 & & \wedge\ s_1 < 0 & \wedge\ s_0 > 0 & & & & & \wedge\ c_1 \geq 0 & \vee \\
s_4 < 0 & \wedge\ s_3 \geq 0 & & & & \wedge\ c_5 \geq 0 & \wedge\ c_4 \geq 0 & \wedge\ c_3 \geq 0 & & & \vee \\
 & & s_2 < 0 & \wedge\ s_1 \geq 0 & & \wedge\ c_5 \geq 0 & \wedge\ c_4 \geq 0 & & \wedge\ c_2 \leq 0 & & \vee \\
 & & & s_1 < 0 & & \wedge\ c_5 > 0 & \wedge\ c_4 > 0 & \wedge\ c_3 > 0 & \wedge\ c_2 \geq 0 & & \vee \\
s_4 < 0 & \wedge\ s_3 > 0 & & & & & & \wedge\ c_3 \geq 0 & \wedge\ c_2 > 0 & \wedge\ c_1 \geq 0 & \vee \\
 & & & s_1 < 0 & \wedge\ s_0 > 0 & & \wedge\ c_4 < 0 & \wedge\ c_3 \leq 0 & & \wedge\ c_1 \geq 0 & \vee \\
s_4 < 0 & \wedge\ s_3 \geq 0 & & & & & \wedge\ c_4 \leq 0 & & \wedge\ c_2 \geq 0 & \wedge\ c_1 \geq 0 & \vee \\
 & s_3 < 0 & \wedge\ s_2 \geq 0 & \wedge\ s_1 \leq 0 & & & & \wedge\ c_3 \geq 0 & \wedge\ c_2 > 0 & & \vee \\
s_4 < 0 & \wedge\ s_3 > 0 & \wedge\ s_2 < 0 & \wedge\ s_1 > 0 & & & & & & \wedge\ c_1 \geq 0 & \vee \\
 & & & s_1 < 0 & \wedge\ s_0 > 0 & \wedge\ c_5 \geq 0 & \wedge\ c_4 \geq 0 & & & \wedge\ c_1 \geq 0 & \vee \\
 & s_3 < 0 & & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & \wedge\ c_4 \leq 0 & & & \wedge\ c_1 \leq 0 & \vee \\
 & s_3 < 0 & \wedge\ s_2 \geq 0 & & & & \wedge\ c_4 < 0 & \wedge\ c_3 < 0 & \wedge\ c_2 \leq 0 & & \vee \\
 & & s_2 \leq 0 & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & & & \wedge\ c_2 \leq 0 & \wedge\ c_1 < 0 & \vee \\
 & s_3 < 0 & \wedge\ s_2 > 0 & \wedge\ s_1 \leq 0 & & & \wedge\ c_4 < 0 & & \wedge\ c_2 \geq 0 & & \vee \\
 & & s_2 < 0 & \wedge\ s_1 > 0 & & \wedge\ c_5 > 0 & & \wedge\ c_3 \leq 0 & & \wedge\ c_1 \geq 0 & \vee \\
 & s_3 < 0 & \wedge\ s_2 \geq 0 & \wedge\ s_1 < 0 & & & & & \wedge\ c_2 \leq 0 & \wedge\ c_1 \leq 0 & \vee
\end{array}$$

$$
\begin{array}{rrrrrrrrrrl}
 & s_3 \leq 0 & & & \wedge\ s_0 > 0 & & \wedge\ c_4 \leq 0 & & \wedge\ c_2 \geq 0 & \wedge\ c_1 > 0 & \vee \\
 & s_3 \leq 0 & & \wedge\ s_1 \geq 0 & & & \wedge\ c_4 \leq 0 & & \wedge\ c_2 \geq 0 & \wedge\ c_1 > 0 & \vee \\
s_4 < 0 & \wedge\ s_3 > 0 & \wedge\ s_2 \leq 0 & & & & & \wedge\ c_3 > 0 & & \wedge\ c_1 \leq 0 & \vee \\
 & & & s_1 < 0 & & \wedge\ c_5 \geq 0 & \wedge\ c_4 \geq 0 & & \wedge\ c_2 \leq 0 & \wedge\ c_1 < 0 & \vee \\
 & & s_2 < 0 & \wedge\ s_1 \geq 0 & \wedge\ s_0 < 0 & & \wedge\ c_4 \leq 0 & & & \wedge\ c_1 \leq 0 & \vee \\
s_4 < 0 & & \wedge\ s_2 \geq 0 & \wedge\ s_1 < 0 & & & & \wedge\ c_3 \geq 0 & & \wedge\ c_1 \leq 0 & \vee \\
 & & s_2 \leq 0 & \wedge\ s_1 > 0 & & & \wedge\ c_4 < 0 & \wedge\ c_3 \leq 0 & & \wedge\ c_1 \geq 0 & \vee \\
 & s_3 < 0 & \wedge\ s_2 \geq 0 & & & \wedge\ c_5 \geq 0 & \wedge\ c_4 \geq 0 & & \wedge\ c_2 \leq 0 & & \vee \\
 & & s_2 \leq 0 & \wedge\ s_1 \geq 0 & & \wedge\ c_5 \geq 0 & \wedge\ c_4 > 0 & \wedge\ c_3 \geq 0 & & & \vee \\
s_4 < 0 & \wedge\ s_3 \geq 0 & & \wedge\ s_1 \leq 0 & & & \wedge\ c_4 \leq 0 & \wedge\ c_3 \leq 0 & & & \vee \\
s_4 \leq 0 & \wedge\ s_3 \geq 0 & & \wedge\ s_1 \leq 0 & & \wedge\ c_5 \geq 0 & \wedge\ c_4 \geq 0 & & & & )
\end{array}
$$

## 7th problem

$$\begin{array}{l}
p_7 > 0 \wedge p_0 > 0 \wedge \quad ( \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_3 < 0 \wedge s_2 > 0 \wedge s_1 < 0 \wedge s_0 > 0 \vee \\
s_4 < 0 \wedge c_6 \geq 0 \wedge c_5 > 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \wedge c_1 < 0 \vee \\
s_5 < 0 \wedge s_4 > 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge c_2 \geq 0 \wedge c_1 > 0 \vee \\
s_5 < 0 \wedge s_4 > 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_2 \geq 0 \wedge c_1 > 0 \vee \\
s_4 \leq 0 \wedge s_3 > 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_6 \geq 0 \wedge c_1 > 0 \vee \\
s_3 \leq 0 \wedge s_2 \geq 0 \wedge c_6 \geq 0 \wedge c_5 > 0 \wedge c_4 \geq 0 \wedge c_1 > 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_6 \geq 0 \wedge c_1 > 0 \vee \\
s_3 < 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_6 \geq 0 \wedge c_1 > 0 \vee \\
s_5 < 0 \wedge s_4 > 0 \wedge s_3 < 0 \wedge s_2 \geq 0 \wedge c_2 > 0 \wedge c_1 \geq 0 \vee \\
s_4 < 0 \wedge s_1 \geq 0 \wedge s_0 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_3 < 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_5 < 0 \wedge s_1 > 0 \wedge s_0 < 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \wedge c_1 < 0 \vee \\
s_1 > 0 \wedge s_0 < 0 \wedge c_6 \geq 0 \wedge c_5 > 0 \wedge c_4 \geq 0 \wedge c_3 \geq 0 \wedge c_1 < 0 \vee \\
s_4 < 0 \wedge s_3 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_5 \leq 0 \wedge c_1 > 0 \vee \\
s_1 > 0 \wedge s_0 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_3 \leq 0 \wedge c_2 \leq 0 \wedge c_1 < 0 \vee \\
s_3 < 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_5 \leq 0 \wedge c_1 > 0 \vee \\
s_3 \geq 0 \wedge s_2 < 0 \wedge c_6 < 0 \wedge c_5 = 0 \wedge c_4 \leq 0 \wedge c_3 \geq 0 \wedge c_2 > 0 \vee \\
s_5 < 0 \wedge s_1 > 0 \wedge s_0 < 0 \wedge c_4 \geq 0 \wedge c_2 \leq 0 \wedge c_1 < 0 \vee \\
s_1 \geq 0 \wedge s_0 < 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \wedge c_3 \leq 0 \wedge c_2 \leq 0 \wedge c_1 < 0 \vee \\
s_1 < 0 \wedge s_0 > 0 \wedge c_6 \geq 0 \wedge c_3 > 0 \wedge c_2 \geq 0 \wedge c_1 > 0 \vee \\
s_1 < 0 \wedge s_0 > 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_4 > 0 \wedge c_1 > 0 \vee \\
s_3 < 0 \wedge s_2 \geq 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_3 \leq 0 \wedge c_1 > 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_3 < 0 \wedge s_2 \geq 0 \wedge c_4 \geq 0 \wedge c_1 \geq 0 \vee \\
s_3 < 0 \wedge s_1 \geq 0 \wedge s_0 < 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \wedge c_1 < 0 \vee \\
s_3 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_3 \geq 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge s_0 < 0 \wedge c_4 \geq 0 \wedge c_3 \geq 0 \wedge c_1 < 0 \vee \\
s_5 < 0 \wedge s_1 > 0 \wedge s_0 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_1 < 0 \vee \\
s_4 < 0 \wedge s_1 \geq 0 \wedge s_0 < 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \wedge c_1 < 0 \vee \\
s_2 < 0 \wedge s_1 > 0 \wedge c_6 \geq 0 \wedge c_3 > 0 \wedge c_2 > 0 \wedge c_1 \geq 0 \vee \\
s_4 < 0 \wedge s_3 > 0 \wedge s_1 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_3 \geq 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_4 \geq 0 \wedge c_3 \geq 0 \vee \\
s_3 \geq 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \wedge c_3 \leq 0 \wedge c_2 \leq 0 \wedge c_1 \geq 0 \vee \\
s_3 < 0 \wedge s_1 \geq 0 \wedge s_0 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_1 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_4 > 0 \wedge c_3 > 0 \wedge c_1 < 0 \vee \\
s_4 < 0 \wedge s_1 > 0 \wedge s_0 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_1 < 0 \vee \\
s_5 < 0 \wedge s_4 > 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_5 \leq 0 \wedge c_1 \geq 0 \vee \\
s_3 < 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_4 \geq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_3 > 0 \wedge s_1 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_5 \geq 0 \wedge s_4 < 0 \wedge s_3 \geq 0 \wedge s_1 < 0 \wedge c_4 \geq 0 \wedge c_3 \geq 0 \wedge c_1 < 0 \vee \\
s_2 < 0 \wedge s_1 > 0 \wedge c_5 \leq 0 \wedge c_4 < 0 \wedge c_2 > 0 \wedge c_1 \geq 0 \vee \\
s_4 < 0 \wedge s_3 \geq 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \vee \\
s_2 < 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_2 < 0 \wedge s_1 \geq 0 \wedge s_0 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \vee \\
s_1 < 0 \wedge s_0 > 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \wedge c_2 > 0 \wedge c_1 > 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_4 \geq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \vee \\
s_1 > 0 \wedge c_5 \leq 0 \wedge c_4 \geq 0 \wedge c_3 = 0 \wedge c_2 > 0 \wedge c_1 \geq 0 \vee
\end{array}$$

$$
\begin{array}{l}
s_3 < 0 \wedge s_2 > 0 \wedge s_1 < 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \wedge c_1 < 0 \vee \\
s_3 < 0 \wedge s_1 > 0 \wedge s_0 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_1 \leq 0 \vee \\
s_3 < 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_3 \leq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_3 < 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge c_1 \leq 0 \vee \\
s_5 < 0 \wedge s_2 \geq 0 \wedge s_0 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_2 \geq 0 \wedge s_1 < 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_3 \leq 0 \wedge c_2 \leq 0 \wedge c_1 < 0 \vee \\
s_4 < 0 \wedge s_3 \geq 0 \wedge s_2 < 0 \wedge s_1 > 0 \wedge c_2 \geq 0 \wedge c_1 > 0 \vee \\
s_4 < 0 \wedge s_3 \geq 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge s_0 < 0 \wedge c_1 \leq 0 \vee \\
s_4 < 0 \wedge s_3 > 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_3 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 \leq 0 \vee \\
s_1 \geq 0 \wedge s_0 < 0 \wedge c_5 \leq 0 \wedge c_4 \geq 0 \wedge c_3 = 0 \wedge c_2 > 0 \vee \\
s_3 < 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_2 \geq 0 \wedge c_1 > 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_3 < 0 \wedge s_2 > 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \vee \\
s_5 < 0 \wedge s_1 > 0 \wedge s_0 < 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \wedge c_1 < 0 \vee \\
s_2 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 \leq 0 \vee \\
s_5 < 0 \wedge s_3 \geq 0 \wedge s_2 < 0 \wedge c_4 \geq 0 \wedge c_2 \leq 0 \wedge c_1 < 0 \vee \\
s_2 > 0 \wedge c_5 < 0 \wedge c_4 = 0 \wedge c_3 \leq 0 \wedge c_2 \geq 0 \wedge c_1 \geq 0 \vee \\
s_4 \leq 0 \wedge s_3 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_4 \geq 0 \wedge c_2 < 0 \vee \\
s_5 \geq 0 \wedge s_4 < 0 \wedge s_3 \geq 0 \wedge s_0 < 0 \wedge c_4 \geq 0 \wedge c_3 \geq 0 \wedge c_1 \leq 0 \vee \\
s_2 \leq 0 \wedge s_1 > 0 \wedge s_0 < 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_1 \leq 0 \vee \\
s_4 < 0 \wedge s_3 \geq 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \vee \\
s_5 < 0 \wedge s_4 > 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \vee \\
s_4 < 0 \wedge s_3 \geq 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \vee \\
s_3 \leq 0 \wedge s_2 \geq 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_2 > 0 \wedge c_1 \geq 0 \vee \\
s_2 < 0 \wedge s_1 > 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \wedge c_2 > 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_4 > 0 \wedge s_3 < 0 \wedge s_2 \geq 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \vee \\
s_3 < 0 \wedge s_2 > 0 \wedge s_0 < 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \wedge c_1 \leq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge s_0 < 0 \wedge c_1 \leq 0 \vee \\
s_4 < 0 \wedge s_3 > 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_3 \leq 0 \wedge c_1 \geq 0 \vee \\
s_1 < 0 \wedge s_0 > 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_3 \leq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_4 > 0 \wedge s_3 < 0 \wedge s_2 \geq 0 \wedge s_0 < 0 \wedge c_1 \leq 0 \vee \\
c_6 \geq 0 \wedge c_5 > 0 \wedge c_4 > 0 \wedge c_3 > 0 \wedge c_2 \geq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge c_4 > 0 \wedge c_3 > 0 \wedge c_2 \geq 0 \wedge c_1 \geq 0 \vee \\
s_4 < 0 \wedge s_3 > 0 \wedge s_1 < 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \wedge c_1 \leq 0 \vee \\
s_5 < 0 \wedge s_3 > 0 \wedge s_2 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 \leq 0 \vee \\
s_3 < 0 \wedge s_2 \geq 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \wedge c_2 > 0 \wedge c_1 \geq 0 \vee \\
s_3 < 0 \wedge s_1 \geq 0 \wedge c_5 < 0 \wedge c_4 < 0 \wedge c_3 \leq 0 \wedge c_2 \leq 0 \wedge c_1 \geq 0 \vee \\
s_4 \leq 0 \wedge s_3 \geq 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_2 \geq 0 \wedge c_1 > 0 \vee \\
s_1 < 0 \wedge s_0 > 0 \wedge c_5 \leq 0 \wedge c_4 \leq 0 \wedge c_3 \geq 0 \wedge c_2 \geq 0 \wedge c_1 > 0 \vee \\
s_5 < 0 \wedge s_4 > 0 \wedge s_1 < 0 \wedge s_0 > 0 \wedge c_3 \leq 0 \wedge c_1 \geq 0 \vee \\
s_4 < 0 \wedge s_3 > 0 \wedge s_1 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_1 \leq 0 \vee \\
s_3 < 0 \wedge s_1 \geq 0 \wedge s_0 < 0 \wedge c_4 \geq 0 \wedge c_3 \geq 0 \wedge c_2 \leq 0 \wedge c_1 \leq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_3 < 0 \wedge s_2 \geq 0 \wedge c_3 \leq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_3 \geq 0 \wedge s_1 < 0 \wedge c_4 > 0 \wedge c_3 \geq 0 \wedge c_1 \leq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_1 < 0 \wedge c_6 \geq 0 \wedge c_5 > 0 \wedge c_1 \leq 0 \vee \\
s_3 \leq 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_4 < 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge c_4 \geq 0 \wedge c_2 \leq 0 \wedge c_1 < 0 \vee \\
s_1 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_3 \leq 0 \wedge c_2 < 0 \wedge c_1 < 0 \vee \\
s_1 < 0 \wedge c_6 \geq 0 \wedge c_5 > 0 \wedge c_4 \geq 0 \wedge c_3 \geq 0 \wedge c_2 \leq 0 \wedge c_1 \leq 0 \vee
\end{array}
$$

$$
\begin{array}{l}
s_4 < 0 \wedge s_3 > 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \wedge c_2 \geq 0 \wedge c_1 \geq 0 \vee \\
s_3 \leq 0 \wedge s_1 \geq 0 \wedge c_6 > 0 \wedge c_5 > 0 \wedge c_4 > 0 \wedge c_3 \geq 0 \vee \\
s_4 < 0 \wedge s_3 \geq 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \wedge c_2 \geq 0 \wedge c_1 \geq 0 \vee \\
s_4 < 0 \wedge s_3 > 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_2 \geq 0 \wedge c_1 \geq 0 \vee \\
s_2 \leq 0 \wedge s_1 > 0 \wedge s_0 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_1 \leq 0 \vee \\
s_4 \leq 0 \wedge s_2 \geq 0 \wedge s_1 \leq 0 \wedge c_5 \leq 0 \wedge c_3 \geq 0 \wedge c_2 > 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_2 < 0 \wedge s_1 \geq 0 \wedge c_4 \geq 0 \wedge c_3 > 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_2 < 0 \wedge s_1 > 0 \wedge c_3 \leq 0 \wedge c_1 \geq 0 \vee \\
s_2 < 0 \wedge s_1 > 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_3 \leq 0 \wedge c_1 \geq 0 \vee \\
s_3 \leq 0 \wedge s_2 \geq 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_2 > 0 \wedge c_1 \geq 0 \vee \\
s_3 \leq 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge s_0 \geq 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \wedge c_1 \leq 0 \vee \\
s_4 \leq 0 \wedge s_3 \geq 0 \wedge c_6 \geq 0 \wedge c_5 > 0 \wedge c_4 \geq 0 \wedge c_2 \leq 0 \vee \\
s_1 < 0 \wedge s_0 > 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_3 \leq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_3 \leq 0 \wedge s_2 > 0 \wedge c_3 \leq 0 \wedge c_2 \geq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge s_1 < 0 \wedge c_5 \leq 0 \wedge c_4 \leq 0 \wedge c_1 < 0 \vee \\
s_3 < 0 \wedge s_2 \geq 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_3 \leq 0 \wedge c_1 \geq 0 \vee \\
s_4 < 0 \wedge s_3 \geq 0 \wedge s_2 \leq 0 \wedge s_1 \geq 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \vee \\
s_4 < 0 \wedge s_3 \geq 0 \wedge s_2 \leq 0 \wedge s_1 \geq 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge c_5 < 0 \wedge c_4 \leq 0 \wedge c_2 \geq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_4 > 0 \wedge s_3 \leq 0 \wedge c_4 \geq 0 \wedge c_3 \geq 0 \wedge c_1 \leq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_2 \geq 0 \wedge c_1 \geq 0 \vee \\
s_5 < 0 \wedge s_4 \geq 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_3 \leq 0 \wedge c_2 \leq 0 \vee \\
s_3 \leq 0 \wedge s_2 \geq 0 \wedge s_1 < 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 \wedge c_1 \leq 0 \vee \\
s_5 \leq 0 \wedge s_4 \geq 0 \wedge s_2 \leq 0 \wedge s_1 \geq 0 \wedge c_6 \geq 0 \wedge c_5 \geq 0 )
\end{array}
$$

## 3.6 Conclusion

This chapter has considered the SDC problem, which is the QE problem $\forall x(x \geq 0 \rightarrow f(x) > 0)$ where $f$ is a polynomial in $\mathbb{R}[x]$. To improve the algorithm, simplification of formulae is important. To simplify a formula, we have shown a necessary condition for the SDC and have used a logic minimization method. Finally, we have shown the effect of our approach by computational results.

We expect to make further reductions to $\varphi_n$ and would like to find necessary and sufficient conditions for the SDC in our future work. Meanwhile, we did not obtain an exact solution to the seventh and subsequent SDC problems. To obtain a simpler formula, we would like to try to simplify formulae by an exact method based, for example, on BDDs.

```
.i 8
.o 1
.lib x0 y0 x1 y1 x2 y2 x3 y3
.ob f0
01010101 1
01010001 1
01011001 1
01011000 1
01011010 1
10010101 1
10010001 1
10011001 1
10011000 1
10011010 1
01000101 1
01000001 1
01001001 1
01001010 2
00001000 1
10001010 1
01100101 1
01100001 1
01101001 1
01100100 1
01100110 1
01101010 2
10101010 1
11------ 2
--11---- 2
----11-- 2
------11 2
0010---- 2
0001---- 2
00----1- 2
00-----1 2
--101000 2
--010100 2
----0010 2
----0000 2
1000--00 2
0100--00 2
000000-- 2
.e
```

```
.lib x0 y0 x1 y1 x2 y2 x3 y3
.i 8
.o 1
.ob f0
.p 4
-11----- 1
-1-----1 1
---1---1 1
-----0-0 1
.e
```

Figure 3.2: Input (left side) and output (right side) file for the 3rd SDC problem for the ESPRESSO command

# Chapter 4

# Symbolic-Numeric Approach to Polynomial Optimization Problems

## 4.1 Introduction

Model-based design has recently attracted attention in manufacturing design, in which a problem can often be specified by mathematical constraints on the mathematical model of a target system. Consequently, developments in design processes are often dependent on the available computational methods — in particular, optimization methods. Numerical convex optimization methods provide globally optimal solutions to many design problems that cannot currently be solved analytically. Design methods based on numerical optimization are becoming more practical because of enhanced computer performance and the development of algorithms with superior accuracy and efficiency. However, hurdles remain with such numerical-computing design methods. To meet manufacturing demands for higher quality, better performance, higher added value, and smaller batches of a variety of products, will require more accurate globally optimal solutions of non-convex problems. At the same time, parametric solutions to problems, such as regions of feasible solutions and optimal solutions in terms of decision variables, will be required.

Constraint solving and optimization methods based on symbolic and algebraic computation have gained attention recently. Specifically, *quantifier elimination* (QE), an algebraic algorithm based on theories of real algebraic geometry, has been successfully applied in many fields of science and engineering [4, 76, 88]. However, to realize practical and effective methods using QE, speed is a significant issue.

In this chapter we consider some important classes of optimization problems that originate from an optimal design process for boosting the yield rate of *static random access memory* (SRAM) products. Numerical approaches based on Monte Carlo methods and *genetic algorithms* (GAs) have been commonly used for such optimization problems. However, more efficient methods that take a broader view of design, encompassing the miniaturization of SRAM technologies, are needed. To that end, we propose new optimization methods based on quantifier elimination combined with numerical computation for optimization problems in which some of the objective functions depend on some of the decision variables.

The organization of the rest of this chapter is as follows: Section 4.2 explains *multi-objective optimization.* Section 4.3 discusses conventional numerical optimization approaches. Our approach to multi-objective optimization is introduced in Section 4.4. Section 4.5 is devoted to demonstrating our methods with concrete computational examples. Section 4.6 concludes this chapter.

## 4.2 Optimization

A typical case of optimization in manufacturing requires engineers to design a product to satisfy given specifications. For each requirement, a performance-measuring *objective function*, with respect to a given set of *decision variables*, is determined. The design problem is to optimize the set of objective functions.

Designers normally draft an initial design for the product using computer aided design software. They then establish some geometric quantities of the product as the decision variables. After assigning a value to each variable, they evaluate the performance of the resulting design with computer simulation. An optimization program can be used to try various combinations of decision values to find a good design.

In the real world, a design that satisfies some requirements might, at the same time, not satisfy others. These types of problems require *multi-objective optimization* (MOO). At some point in MOO, one can improve an objective function value only at the expense of another. This problem is typical in MOO, not occurring in the case of *single-objective optimization* (SOO). The main source of difficulty is the fact that the objective space cannot be totally ordered canonically.

In the rest of this section we define some terms and formally state the problem.

### 4.2.1 Multi-Objective Optimization

Here we give a formal setting of an MOO problem [62].

We denote the real number field by $\mathbb{R}$. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a vector of decision variables, and $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x}),\ \ldots,\ f_r(\boldsymbol{x}))$ be a vector of real-valued functions. These functions are called objective functions, to be optimized in an MOO problem. Decision variable values are real numbers. Suppose that the vector of decision variables $\boldsymbol{x}$ varies through a prescribed semialgebraic set $\mathcal{P} \subseteq \mathbb{R}^n$ of the $n$-dimensional Euclidean space, defined by a finite family of equations, inequations, and inequalities, or a finite union of such sets. Call $\mathcal{P}$ a *feasible region in the decision space.* The image $\mathcal{F} = \{\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^r | \boldsymbol{x} \in \mathcal{P}\}$ of $\mathcal{P}$ under $\boldsymbol{f}$ is called the *feasible region in the objective space.*

Assume that a lower function value means higher performance. An MOO problem can be formulated as follows:

$$\begin{cases} \text{Minimize} & \boldsymbol{f}(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{P}. \end{cases}$$

We define an order in the objective space to specify what minimize means in the problem.

**Definition 57.** *Let $r$ be a positive integer, and $\boldsymbol{a} = (a_1, \ldots, a_r)$ and $\boldsymbol{b} = (b_1, \ldots, b_r)$ points in $\mathbb{R}^r$. We say $\boldsymbol{a}$* dominates *$\boldsymbol{b}$ when $a_i \leq b_i$ for $i = 1, \ldots, r$. Denote $\boldsymbol{a} \preceq \boldsymbol{b}$ to represent $\boldsymbol{a}$ dominates $\boldsymbol{b}$.*

Thus $\mathbb{R}^r$ is a *partially ordered set* with respect to $\preceq$. Note that it is a total ordering only when $r = 1$ and in that case the relation $\preceq$ meets the usual weak inequality $\leq$ used in SOO, i.e., $\boldsymbol{x}_0 \in \mathcal{P}$ is optimal when

$$\boldsymbol{f}(\boldsymbol{x}_0) \leq \boldsymbol{f}(\boldsymbol{x})\ ,\ \forall\ \boldsymbol{x} \in \mathcal{P}$$

holds. If there exists $\boldsymbol{x}_0$ satisfying a similar condition

$$\boldsymbol{f}(\boldsymbol{x}_0) \preceq \boldsymbol{f}(\boldsymbol{x})\ ,\ \forall\ \boldsymbol{x} \in \mathcal{P}$$

in an MOO problem, $\boldsymbol{x}_0$ is called an *absolutely optimal solution.* However, in most real problems there does not exist an absolutely optimal solution because there are trade-offs among objective functions.

In an MOO problem one must consider a solution as a set rather than a single point.

**Definition 58.** *Let $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_r(\boldsymbol{x}))$ be a vector of objective functions, $\mathcal{P} \subseteq \mathbb{R}^n$ a semialgebraic set, and $\mathcal{F} = \{\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^r | \boldsymbol{x} \in \mathcal{P}\}$ and consider the optimization problem. $\boldsymbol{x}_0 \in \mathcal{P}$ is called a* Pareto optimal solution *if for any $\boldsymbol{x} \in \mathcal{P}$, $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_0)$ or $\boldsymbol{f}(\boldsymbol{x}) \not\preceq \boldsymbol{f}(\boldsymbol{x}_0)$. Such points form a set, called a* Pareto optimal set. *The image of the Pareto optimal set is called the* Pareto optimal front*, which is a subset of the objective space.*

We could define that for two points $\boldsymbol{a}$ and $\boldsymbol{b}$, $\boldsymbol{a}$ *strongly dominates* $\boldsymbol{b}$ if $\boldsymbol{a} \preceq \boldsymbol{b}$ and $\boldsymbol{a} \neq \boldsymbol{b}$. Using this term, $\boldsymbol{x}_0$ is Pareto optimal if no points in $\mathcal{F}$ strongly dominate $\boldsymbol{f}(\boldsymbol{x}_0)$.

### 4.2.2 Parametric Optimization

We consider a parametric optimization problem.

Let $\boldsymbol{x} = (x_1, \dots, x_n)$ be a vector of decision variables, $\boldsymbol{\theta} = (\theta_1, \dots, \theta_t)$ a vector of parameters, and $f_\theta(\boldsymbol{x}) = f(\boldsymbol{\theta}, \boldsymbol{x})$ an objective function. A parametric optimization problem can be formulated as follows:

$$\begin{cases} \text{Minimize} & f_\theta(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{\theta} \in \mathcal{T},\ \boldsymbol{x} \in \mathcal{P}_\theta, \end{cases}$$

where $\mathcal{T} \subseteq \mathbb{R}^t$ and $\mathcal{P}_\theta \subseteq \mathbb{R}^n$ are feasible regions in the parameter space and the decision space, respectively.

The goal of parametric optimization is to obtain the optimal solution as a function on parameters $\boldsymbol{\theta}$.

### 4.2.3 Minimax Optimization

Here we discuss a special type of optimization problem called minimax optimization.

Let $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ be a vector of decision variables, $h_i(\boldsymbol{x})$ a real-valued function for $i = 1, \dots, r$, and $\mathcal{P} \subseteq \mathbb{R}^n$ a feasible region in the decision space. Let $f_{max}(\boldsymbol{x})$ be an objective function defined by $f_{max}(\boldsymbol{x}) \equiv \max(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \dots, h_r(\boldsymbol{x}))$, where function max returns the maximal entry among $h_1(\boldsymbol{x})$, …, $h_r(\boldsymbol{x})$.

A minimax optimization problem can be formulated as follows:

$$\begin{cases} \text{Minimize} & f_{max}(\boldsymbol{x}) \equiv \max(h_1(\boldsymbol{x}), \dots, h_r(\boldsymbol{x})) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{P}. \end{cases} \tag{4.1}$$

Note that this is an SOO problem.

## 4.3 Known Approaches to MOO

Many methods exist for finding solutions to MOO problems. One classical method is to change the problem into an SOO problem and apply an SOO algorithm. Examples are the weighted-sum strategy [90] and the $\epsilon$-constraint method [38]. However, these methods cannot handle MOO problems with a discontinuous and non-convex Pareto optimal front.

This section introduces *evolutionary algorithms* (EAs), the most popular approach to MOO problems. We compare our symbolic-numeric method with two types of EA in Section 4.5.2.

### 4.3.1 Evolutionary Algorithms

EAs are categorized as meta-heuristic search methods, which originated in the principle of natural selection. An EA starts with a group of points in the decision space, called a population; these points, or individuals, move or change after the selection and reproduction processes according to their objective function values. Superior points survive with a higher probability; they also have a better chance to have descendants. Mutation prevents a population from falling into a local optimum. One expects a population to move toward the Pareto optimal front by repeating these processes. For details, refer to [32] for a comprehensive guide to the area. Approaches using EAs have been popular because they work well with parallel computing and because efficient large-scale experiments are realizable.

#### Genetic Algorithms

*Genetic algorithms* (GAs), first proposed by J.H. Holland in 1975, simulate genetic reproduction, crossover (recombination), and mutation [33]. An individual is encoded in a binary string called a gene. A gene is manipulated to produce a new one according to a crossing rule, with an occasional event of mutation. Adopting a different set of rules for encoding or crossing can change the future of the population. Collecting the individuals of an entire generation produces information on the feasible region in the objective space and the Pareto optimal front. But it can be difficult to determine the theoretical meaning of the operations on genes, such as reproduction and crossover. Since genetic algorithms are the most popular type of EA, there exist many implementations—for example, the commercial software Matlab, Optimus, modeFrontier, and Isight.

#### Particle Swarm Optimization

*Particle Swarm Optimization* (PSO) was introduced by R. Eberhart and J. Kennedy [31]. The behavior of a swarm of insects or of a flock of birds are metaphors for this method. Individuals are placed at random in the decision space. Each of them simultaneously moves to the sum of relative vectors from itself to: (1) the group's best position so far, (2) the individual's best position, and (3) a random vector, multiplied by respective constants. A random vector prevents a swarm from falling into a local optimum.

This process constitutes a generation. As time passes, a generation gathers optimal points. It masses at one optimal point or splits into smaller groups, each gathering around a local point. Many variants for PSO have been proposed. Even in the basic algorithm described above it is left to the user to determine the constant values by which the three vectors are multiplied. PSO is used in commercial optimization software such as modeFrontier and Isight.

## 4.4 The Quantifier Elimination Method

In this section, we present the QE-based symbolic algorithm for MOO.

### 4.4.1 The Symbolic Method

The symbolic method, proposed by Yanami [87], begins with a polynomial model for each objective function. In most applications, an optimization process is realized via a simulator that receives a list of real values for decision variables and returns a collection of real values representing various physical properties of the product. The objective functions are computed from these output values.

We need a model that not only fits input-output data well but one that is expressed simply for symbolic computation to work. A low-degree model is desirable for QE.

#### Expressing the Pareto Optimal Front

Once the objective functions are expressed as approximated polynomial models in decision variables, one can formulate a constraint as a first-order formula and compute a Pareto optimal front by QE. To show how an MOO problem is interpreted as a first-order formula, recall that a feasible region $\mathcal{P}$ in the decision space and a feasible region

$\mathcal{F}$ in the objective space of an MOO problem can be expressed as

$$\begin{aligned}\mathcal{P} &= \{\boldsymbol{x} \in \mathbb{R}^n | \varphi_{\mathcal{P}}(\boldsymbol{x})\} \ , \\ \mathcal{F} &= \{\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^r | \boldsymbol{x} \in \mathcal{P}\} \ ,\end{aligned}$$

where $\varphi_{\mathcal{P}}(\boldsymbol{x}) = \varphi_{\mathcal{P}}(x_1, \ldots, x_n)$ is called a *defining formula* for $\mathcal{P}$. It is straightforward to construct $\varphi_{\mathcal{P}}$ when $\mathcal{P}$ is semialgebraic. From these notations one can naturally construct a first-order formula $\psi_{Pareto}$ with free variables $\boldsymbol{y} = (y_1, \ldots, y_r)$ that is true at $\boldsymbol{y} = \boldsymbol{y}_0$ if and only if $\boldsymbol{y}_0$ is on the Pareto optimal front:

$$\psi_{Pareto} = \exists \boldsymbol{x} \forall \boldsymbol{u} \ (\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \wedge \varphi_{\mathcal{P}}(\boldsymbol{x}) \wedge (\varphi_{\mathcal{P}}(\boldsymbol{u}) \rightarrow (\boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{x}) \vee \boldsymbol{f}(\boldsymbol{u}) \not\leq \boldsymbol{f}(\boldsymbol{x})))) \ .$$

By eliminating $\boldsymbol{x}$ and $\boldsymbol{u}$ from $\psi_{Pareto}$ we obtain a quantifier-free formula with respect to $\boldsymbol{y}$. But in our approach it is natural to construct the feasible region in the objective space itself, which includes the information on the Pareto optimal front.

## Expressing the Feasible Region

In our formulation, expressing the entire feasible region in the objective space is easier than expressing the Pareto optimal front. Using the same notations as above, naturally construct a first-order formula $\psi_{Feasible}$ with free variables $\boldsymbol{y} = (y_1, \ldots, y_r)$ that is true at $\boldsymbol{y} = \boldsymbol{y}_0$ if and only if $\boldsymbol{y}_0$ is in the feasible region:

$$\psi_{Feasible} = \exists \boldsymbol{x} \ (\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \wedge \varphi_{\mathcal{P}}(\boldsymbol{x})) \ . \tag{4.2}$$

Obviously $\psi_{Feasible}$, taking away the part that is to remove the dominated points from $\psi_{Pareto}$, is easier to solve than $\psi_{Pareto}$. By eliminating $\boldsymbol{x}$ from $\psi_{Feasible}$ obtain a quantifier-free formula with respect to $\boldsymbol{y}$. This means that QE can compute not only the Pareto optimal front but the exact feasible region, bringing an enormous advantage compared to numerical optimization methods that usually find only one optimal point—or at most a finite set of them—at a time. In this approach, we can see the Pareto optimal front from the plot of the feasible region.

**Example 59.** *The following MOO problem is taken from [25, p. 79].*

$$\begin{cases} \textit{Minimize} & f(x_1, x_2) = x_1^2 + x_2^2 \ \textit{and} \\ \textit{minimize} & g(x_1, x_2) = 5 + x_2^2 - x_1 \\ \textit{subject to} & -5 \leq x_1 \leq 5, \ -5 \leq x_2 \leq 5. \end{cases}$$

*The dimensions of both the decision space and the objective space are two. The associated QE problem is given as:*

$$\exists x_1 \exists x_2 \quad (f = x_1^2 + x_2^2 \ \wedge \ g = 5 + x_2^2 - x_1 \ \wedge \ -5 \le x_1 \le 5 \ \wedge \ -5 \le x_2 \le 5).$$

*Using a QE algorithm we obtain the following exact feasible region in the $f - g$ plane as a semialgebraic set.*

$$\begin{aligned}
&(g - f + 25 \ge 0 \ \wedge \ g^2 - 60g - f + 925 \ge 0 \ \wedge \ g \le 30 \ \wedge \ f \ge 25) \ \vee \\
&(4g - 4f - 21 \le 0 \ \wedge \ g \ge 30 \ \wedge \ 4f \le 101) \ \vee \\
&(g - f + 15 \ge 0 \ \wedge \ g^2 - 60g - f + 925 \le 0) \ \vee \\
&(g - f + 25 \ge 0 \ \wedge \ g^2 - 10g - f + 25 \le 0) \ \vee \\
&(4g - 4f - 21 \le 0 \ \wedge \ g \ge 5 \ \wedge \ f \le 25 \ \wedge \ 4f \ge 1).
\end{aligned}$$

*Figures 4.1 and 4.2 show the results of using the QE algorithm and a GA-based method as a numeric method respectively. Since the shaded part in Figure 4.1 is the exact feasible region in the objective space, we can see the exact Pareto optimal front. On the other hand, we can estimate the Pareto optimal front from Figure 4.2. Since the result of the numerical method is a set of points approximating the entire Pareto optimal front, we are not able to know how close this is to the Pareto optimal front.*

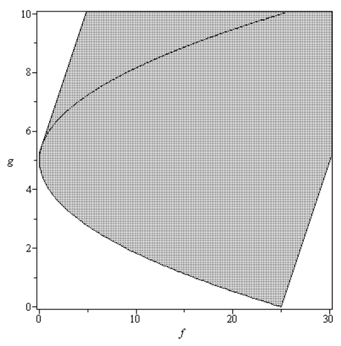

Figure 4.1: Symbolic approach to MOO

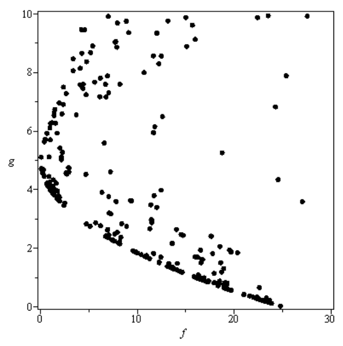

Figure 4.2: Numeric approach to MOO

## A Symbolic Approach to Minimax Optimization Problems

Here we show our symbolic approach to minimax optimization problems. To start, we are not able to formulate a max function as a first-order formula. Using the same notations as above, we construct a first-order formula $\psi_{minimax}$ with free variable $y$ that is true at $y \ge y_0$, where $y_0$ is the minimal value:

$$\psi_{minimax} = \exists \boldsymbol{x} \ (y \ge h_1(\boldsymbol{x}) \wedge \cdots \wedge y \ge h_r(\boldsymbol{x}) \wedge \varphi_{\mathcal{P}}(\boldsymbol{x})) \ . \tag{4.3}$$

By eliminating $\boldsymbol{x}$ from (4.3) we obtain a quantifier-free formula $\phi_{minimax}$ with respect to $y$. The feasible region in the objective space of (4.1) is not the same feasible region $\phi_{minimax}$. However, they do have the same minimal value. Thus we obtain the exact minimal value with this approach.

**Example 60.** *Consider the following minimax optimization problem:*

$$\begin{cases} \textit{Minimize} & f(x) \equiv \max(h_1(x), h_2(x)) \\ \textit{subject to} & h_1(x) \equiv -x^2 - 4x + 1, h_2(x) \equiv -x^2 + 4x + 1, \\ & -3 \leq x \leq 3. \end{cases}$$

*This is an SOO problem; the dimension of the decision space is one. Figure 4.3 shows the graph of polynomials $h_1(x)$ and $h_2(x)$. We can see that the feasible region in the objective space is expressed as $1 \leq f(x) \leq 5$.*

*Using our approach, solve the following QE problem:*

$$\exists x\ (y \geq -x^2 - 4x + 1\ \wedge\ y \geq -x^2 + 4x + 1\ \wedge\ -3 \leq x \leq 3). \tag{4.4}$$

*Performing QE on (4.4), obtain an equivalent quantifier-free formula $y \geq 1$, which is not equivalent to $1 \leq y \leq 5$, but both have the same minimal value of $y = 1$.*

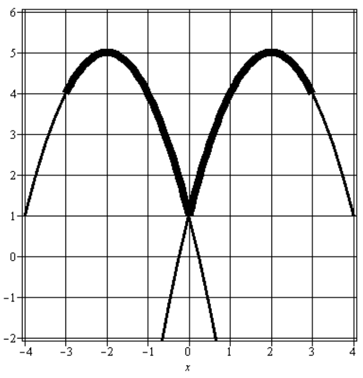

Figure 4.3: Example of minimax problem

## 4.4.2 The Symbolic-Numeric Approach

Here we show our symbolic-numeric approach to MOO problems. Although numerical methods for MOO problems are effective, it is difficult to know how many iterations are sufficient to guarantee a certain precision. On the other hand, the symbolic approach produces an exact Pareto optimal front for an MOO problem, even if it is non-convex.

Although QE is a powerful tool for MOO problems, it is computationally very hard; Davenport and Heinz [24] proved that the worst-case computational complexity is doubly exponential in the number of quantified variables. Consequently, the elimination $\boldsymbol{x}$ from $\psi_{Feasible}$ may not terminate in a reasonable time, or terminate with an error. To address this issue, we propose a new combined method of QE and a numerical method.

Consider the case where there exist integers $m < n$ and $s \le r$ such that $f_i$ depends only on $(x_1, \dots, x_m)$ for $i = 1, \dots, s$. In this case, one can subdivide the optimization problem and solve the sub-problem with our symbolic approach.

Let $\boldsymbol{x}_M = (x_1, \dots, x_m)$ be an $m$-dimensional vector, $\boldsymbol{x}_N = (x_{m+1}, \dots, x_n)$ an $(n-m)$-dimensional vector, $\boldsymbol{f}_S(\boldsymbol{x}_M) = (f_1(\boldsymbol{x}_M), \dots, f_s(\boldsymbol{x}_M))$ an $s$-dimensional vector, $\boldsymbol{f}_R(\boldsymbol{x}_M, \boldsymbol{x}_N) = (f_{s+1}(\boldsymbol{x}), \dots, f_r(\boldsymbol{x}))$ an $(r-s)$-dimensional vector, $\boldsymbol{y}_S = (y_1, \dots, y_s)$ an $s$-dimensional vector, and $\boldsymbol{y}_R = (y_{s+1}, \dots, y_r)$ an $(r-s)$-dimensional vector.

Reformulate (4.2) as follows:

$$\begin{aligned}\psi_{Feasible} &= \exists \boldsymbol{x}_M \exists \boldsymbol{x}_N (\boldsymbol{y}_S = \boldsymbol{f}_S(\boldsymbol{x}_M) \wedge \boldsymbol{y}_R = \boldsymbol{f}_R(\boldsymbol{x}_M, \boldsymbol{x}_N) \wedge \varphi_{\mathcal{P}}(\boldsymbol{x}_M, \boldsymbol{x}_N)) \\ &= \exists \boldsymbol{x}_M \ (\boldsymbol{y}_S = \boldsymbol{f}_S(\boldsymbol{x}_M) \wedge \exists \boldsymbol{x}_N \ (\boldsymbol{y}_R = \boldsymbol{f}_R(\boldsymbol{x}_M, \boldsymbol{x}_N) \wedge \varphi_{\mathcal{P}}(\boldsymbol{x}_M, \boldsymbol{x}_N))).\end{aligned}$$

Obtain the sub-problem:

$$\exists \boldsymbol{x}_N \ (\boldsymbol{y}_R = \boldsymbol{f}_R(\boldsymbol{x}_M, \boldsymbol{x}_N) \wedge \varphi_{\mathcal{P}}(\boldsymbol{x}_M, \boldsymbol{x}_N)). \tag{4.5}$$

Performing QE on (4.5), obtain an equivalent quantifier-free formula $\psi_{\mathcal{P}}(\boldsymbol{x}_M, \boldsymbol{y}_R)$. Clearly (4.5) is easier to solve than $\psi_{Feasible}$. Finally, solve the following MOO problem, equivalent to the original one, using a numerical method:

$$\begin{cases} \text{Minimize} & \boldsymbol{y}_S = \boldsymbol{f}_S(\boldsymbol{x}_M) \text{ and} \\ \text{minimize} & \boldsymbol{y}_R \\ \text{subject to} & \psi_{\mathcal{P}}(\boldsymbol{x}_M, \boldsymbol{y}_R) \ . \end{cases} \tag{4.6}$$

In this problem the decision variables are $x_1, \dots, x_m$ and $y_{s+1}, \dots, y_r$. When the number of decision variables $m + r - s$ is less than $n$, our symbolic-numeric approach makes numerical optimization computations efficient and precise.

Another advantage of numerical methods is that fact that one can apply them directly for any objective functions that are not expressed as polynomials. However, with the symbolic-numeric approach, we might encounter non-polynomial objective functions. Section 4.5.2 contains an example in which an objective function is exponential.

## 4.5 Application

This section describes how our methods are applied to actual design problems in manufacturing. We specify a set of example problems and show computational results for each.

### 4.5.1 Problem Statements

Our target problems derive from SRAM optimal design. Figure 4.4 shows a scanning electron microscope (SEM) picture of an SRAM cell.

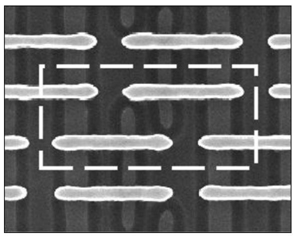

Figure 4.4: SEM picture of SRAM cell

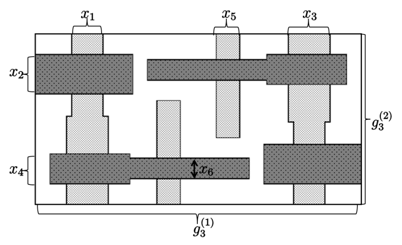

Figure 4.5: SRAM schematic diagram

Figure 4.5 shows a schematic diagram of an SRAM cell of the dotted rectangle part of Figure 4.4. It is required to design an SRAM cell of small size but with a high yield rate, which are, of course, conflicting requirements.

We design the layout of an SRAM cell by assigning real values to design variables. There are six such variables, each of which corresponds to a channel length or channel width. Denote these design variables by $\boldsymbol{x} = (x_1, \ldots, x_6)$. Each variable runs through a designated interval $I_i = [a_i, b_i]$, $a_i, b_i \in \mathbb{R}$, $i = 1, \ldots, 6$. Thus we have $\mathcal{P} = [a_1, b_1] \times \cdots \times [a_6, b_6] \subseteq \mathbb{R}^6$ as a design space.

To evaluate the quality of an SRAM cell, two types of estimators, each on a noise margin, are used. These are functions on the design variables $\boldsymbol{x}$. We call them $g_1$ and $g_2$, both linear functions on $\boldsymbol{x}$. We prepare two variables $y_1$ and $y_2$ for these estimators and write $y_1 \equiv g_1(\boldsymbol{x})$ and $y_2 \equiv g_2(\boldsymbol{x})$. The yield rate $z$, the most important objective function in the design, is expressed as $z \equiv \min(y_1, y_2)$. Note that the yield rate is expressed in sigma value, i.e., a deviation from the mean of a standard normal distribution. Expressions using sigma are often used for an event occurring with very low probability. Statistically, the higher the minimum of them, the higher the yield rate.

For the SRAM cell use $g_3^{(1)}$ for its width and $g_3^{(2)}$ for its length. Both are linear functions on the design variables $\boldsymbol{x}$. The cell is viewed as a rectangle; its size $g_3$ is expressed as $g_3 = g_3^{(1)} \cdot g_3^{(2)}$. Prepare $y_3$ for the area and write $y_3 \equiv g_3(\boldsymbol{x})$.

Another objective function $y_4 \equiv g_4(\boldsymbol{x})$ on a leakage current is considered occasionally. The function $g_4$ is exponential with $\boldsymbol{x}$, the only non-polynomial objective function in the problems.

To summarize, it is necessary to solve the specified problem by maximizing the yield rate $z$ and minimizing the cell size $y_3$, or a leakage current $y_4$. This problem is one of a class of optimization problems such as parametric optimization and multi-objective optimization. Fundamental problems encountered in the design process, which are our main concern here, are formulated below. Although we do not discuss combined problems, such problems are frequently encountered in actual design work. Note that in Problem 2 we use a design variable $x_2$ as a parameter.

**Problem 1** Find the maximal value of the yield rate $z$:

$$\begin{cases} \text{Maximize} & z \equiv \min(y_1, y_2) \\ \text{subject to} & y_1 \equiv g_1(\boldsymbol{x}),\ y_2 \equiv g_2(\boldsymbol{x}),\ \ \boldsymbol{x} \in \mathcal{P} \subseteq \mathbb{R}^6. \end{cases}$$

**Problem 2** Find the relation between the yield rate $z$ and a design parameter $x_2$.

**Problem 3** Find the relation between the yield rate $z$ and the cell size $y_3$.

**Problem 4** Find the relation between the yield rate $z$ and the objective function $y_4$ on a leakage current, which is exponential on $\boldsymbol{x}$.

In general, the functions for noise margins $g_1$, $g_2$ and the function for the cell size $g_3$ are nonlinear with respect to the design variables $\boldsymbol{x}$. However, note that one can expect some special structures in the optimization problems derived from the actual circumstances of SRAM optimal design. In fact, one can assume the following properties for most problems of SRAM optimal design. Our aim is to develop effective and efficient algorithms to solve Problems 1, 2, 3, and 4 by exploiting the following properties.

**Structure 1** $g_1$ and $g_2$ are polynomial models generated from simulation data. In many cases, it is sufficient to employ linear models with respect to $\boldsymbol{x}$ for $g_1$ and $g_2$.

**Structure 2** The objective function for the cell size $g_3$ is factored into its width $g_3^{(1)}$ and its length $g_3^{(2)}$, each of which is linear with respect to $\boldsymbol{x}$: $g_3(\boldsymbol{x}) = g_3^{(1)}(\boldsymbol{x}) \cdot g_3^{(2)}(\boldsymbol{x})$.

**Structure 3** Although the objective function $g_4$ associated with a leakage current is exponential, only two design variables out of six are involved: $g_4 = g_4(x_1, x_2)$.

### 4.5.2 Computational Results

We briefly show computational results for all four problems. All examples appearing in this subsection are derived from actual SRAM design processes in our company.

We solve the example problems using SyNRAC [88] and REDLOG [28] on a PC with a 1.60 GHz CPU and 8.0 GB of memory. A genetic algorithm (GA) [33] and PSO [31] are used as conventional numerical methods. For a GA tool, we used the *Single and Multiobjective Genetic Algorithm Toolbox* [70], developed by K. Sastry. For a PSO tool, we used *modeFrontier 4.2.1.*

#### Problem 1

Here we show an example of our symbolic approach to an SOO problem. The concrete problem is as follows:

$$\begin{cases} \text{Maximize} & z \equiv \min(y_1, y_2) \\ \text{subject to} & y_1 \equiv g_1(\boldsymbol{x}),\, y_2 \equiv g_2(\boldsymbol{x}), \\ & 0 \le x_1 \le 1,\, \ldots,\, 0 \le x_6 \le 1,\, x_4 + x_6 \ge 2x_2, \end{cases} \tag{4.7}$$

where

$$\begin{aligned} g_1(\boldsymbol{x}) &= 4.34758037607255 + 0.215813228985934x_1 - 0.402110351083682x_2 \\ &\quad +2.76367763462092x_3 + 0.472650590690848x_4 - 0.291960906981533x_5 \\ &\quad +1.48362647919883x_6 \end{aligned}$$

and

$$\begin{aligned} g_2(\boldsymbol{x}) &= 2.55801233493670 - 0.245208280326772x_1 + 1.13856413840377x_2 \\ &\quad -0.219401355823440x_3 + 0.0882262731070385x_4 + 1.75046245313323x_5 \\ &\quad -0.615878109869250x_6. \end{aligned}$$

Note that the noise margins $g_1$ and $g_2$ are linear in $\boldsymbol{x}$.

Since we are not able to formulate the min function as a first-order formula, we apply the approach to minimax optimization problems shown in Section 4.4.1. The associated QE problem is given as

$$\begin{aligned}\exists x_1 \cdots \exists x_6 \quad (z \le g_1(\boldsymbol{x}) \ \wedge \ z \le g_2(\boldsymbol{x}) \ \wedge \\ 0 \le x_1 \le 1 \ \wedge \cdots \wedge \ 0 \le x_6 \le 1 \ \wedge \ x_4 + x_6 \ge 2x_2).\end{aligned} \tag{4.8}$$

The polynomials appearing in (4.8) are all linear. Consequently, we can use a specialized QE algorithm [81], which is normally more efficient than a general one. Performing QE on (4.8), produces an equivalent quantifier-free formula describing the feasible region for (4.8).

$$265833468864870800000000000000z \le 1314171707638219319422773397743.$$

Obtain the exact maximal value

$$z_{max} = \frac{1314171707638219319422773397743}{265833468864870800000000000000} \simeq 4.9435901.$$

**Remark 61.** *The coefficients of $g_1(\boldsymbol{x})$ and $g_2(\boldsymbol{x})$ are expressed as floating point numbers. Solving an optimization problem with QE, one converts floating point numbers to rational numbers and computes over the rational number field.*

**Problem 2**

Here we show an example of our symbolic approach to a parametric optimization problem. This framework is similar to that of an MOO problem. The concrete parametric optimization problem is as follows:

$$\begin{cases} \text{Maximize} & z \equiv \min(y_1, y_2) \\ \text{subject to} & y_1 \equiv g_1(\boldsymbol{x}),\ y_2 \equiv g_2(\boldsymbol{x}), \\ & 0 \le x_1 \le 1,\ \ldots,\ 0 \le x_6 \le 1,\ \ x_4 + x_6 \ge 2x_2. \end{cases}$$

The associated QE problem is given as

$$\begin{aligned}\exists x_1 \exists x_3 \exists x_4 \exists x_5 \exists x_6 \quad (z \le g_1(\boldsymbol{x}) \ \wedge \ z \le g_2(\boldsymbol{x}) \ \wedge \\ 0 \le x_1 \le 1 \ \wedge \cdots \wedge \ 0 \le x_6 \le 1 \ \wedge \ x_4 + x_6 \ge 2x_2).\end{aligned} \tag{4.9}$$

The difference between (4.8) and (4.9) is the quantification of $x_2$. The polynomials appearing in (4.9) are also all linear; so one can utilize a specialized QE algorithm [81]. Performing QE on (4.9), obtain an equivalent quantifier-free formula $\phi(z, x_2)$, which shows the feasible region in the $x_2$-$z$ plane as a semialgebraic set.

$$\begin{aligned}
&\phi(z, x_2) = 0 \leq x_2 \leq 1 \ \wedge \\
&2000000000000000z - 2277128276807540x_2 - 8793402122353937 \leq 0 \ \wedge \\
&1491539495222180000000000000000z - 1529200344334768779206195881 16x_2 \\
&-6572286489769842032866193098 31 \leq 0 \ \wedge \\
&1491539495222180000000000000000z - 1526214475082968583178157463 6x_2 \\
&-7260575938183077993310383165 71 \leq 0 \ \wedge \\
&2000000000000000z + 186384162669460x_2 - 10025158342092437 \leq 0.
\end{aligned}$$

The shaded part in Figure 4.6 shows the feasible region given by the above formula $\phi(z, x_2)$ in the $x_2$-$z$ plane. By focusing attention on the upper bounds of the feasible region with respect to $z$, if they exist, one can see the explicit relation between the maximal yield rate and the design parameter $x_2$.

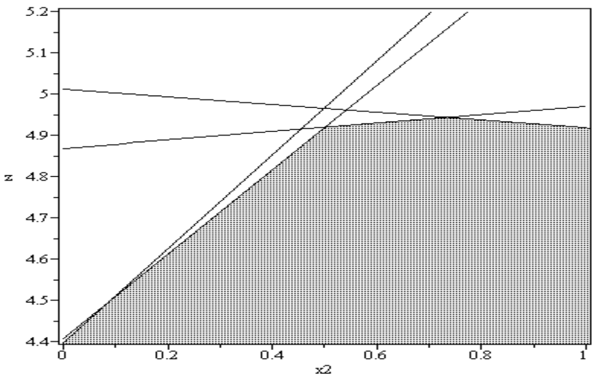

Figure 4.6: Feasible region given by $\phi(z, x_2)$

## Problem 3

Here we show our approach to an MOO problem to find the relation between the yield rate and the cell size, one of the most important objective functions in SRAM optimal design. We cannot obtain the feasible region in the objective space by our symbolic approach due to its computational complexity. Thus we apply our symbolic-numeric approach to the problem. We compare a conventional numerical method with our symbolic-numeric approach. Finally, we show a particular advantage of our symbolic approach.

The concrete MOO problem is as follows:

$$\begin{cases} \text{Maximize} & z \equiv \min(y_1, y_2) \text{ and} \\ \text{minimize} & y_3 = y_3^{(1)} \cdot y_3^{(2)} \\ \text{subject to} & y_1 \equiv g_1(\boldsymbol{x}),\ y_2 \equiv g_2(\boldsymbol{x}),\ y_3^{(1)} \equiv g_3^{(1)}(\boldsymbol{x}),\ y_3^{(2)} \equiv g_3^{(2)}(\boldsymbol{x}), \\ & g_3^{(1)}(\boldsymbol{x}) = 1420 + 250x_1 + \max(250x_5, 250x_3 - 60), \\ & g_3^{(2)}(\boldsymbol{x}) = 800 + \max(40x_4 + 40x_6, 140x_2 - 10), \\ & 0 \le x_1 \le 1,\ \ldots,\ 0 \le x_6 \le 1,\ x_4 + x_6 \ge 2x_2. \end{cases} \tag{4.10}$$

The associated QE problem is given as

$$\begin{aligned} \exists x_1 \cdots \exists x_6 \exists y_3^{(1)} \exists y_3^{(2)} \quad & (z \le g_1(\boldsymbol{x}) \ \wedge \ z \le g_2(\boldsymbol{x}) \ \wedge \ y_3 = y_3^{(1)} \cdot y_3^{(2)} \ \wedge \\ & y_3^{(1)} \ge 1420 + 250x_1 + 250x_5 \ \wedge \\ & y_3^{(1)} \ge 1420 + 250x_1 + 250x_3 - 60 \ \wedge \\ & y_3^{(2)} \ge 800 + 40x_4 + 40x_6 \ \wedge \\ & y_3^{(2)} \ge 800 + 140x_2 - 10 \ \wedge \\ & 0 \le x_1 \le 1 \ \wedge \cdots \wedge \ 0 \le x_6 \le 1 \ \wedge \ x_4 + x_6 \ge 2x_2). \end{aligned} \tag{4.11}$$

Performing QE on (4.11), obtain an equivalent quantifier-free formula $\tau(z, y_3)$, which demonstrates the feasible region in the $y_3$-$z$ plane as a semialgebraic set. By focusing on the upper bounds of the feasible region with respect to $z$ and the lower bounds with respect to $y_3$ simultaneously, one can see the trade-off relation between the yield rate $z$ and the cell size $y_3$.

Since the formula (4.11) has a quadratic polynomial $y_3$ in $\boldsymbol{x}$, it might significantly increase the computational complexity. In fact, the computation of QE on (4.11) did not terminate after an hour. Next, apply the symbolic-numeric approach to (4.11). Eliminating as many variables as possible increases the effectiveness of the symbolic-numeric approach. Since the complexity of a QE algorithm depends on the order and the choice of variables, the selection of a list of quantified variables is important.

We chose six variables $x_1, \ldots, x_6$ to eliminate. Consider the following QE problem:

$$\begin{aligned} \exists x_1 \cdots \exists x_6 \quad & (z \le g_1(\boldsymbol{x}) \ \wedge \ z \le g_2(\boldsymbol{x}) \ \wedge \\ & y_3^{(1)} \ge 1420 + 250x_1 + 250x_5 \ \wedge \\ & y_3^{(1)} \ge 1420 + 250x_1 + 250x_3 - 60 \ \wedge \\ & y_3^{(2)} \ge 800 + 40x_4 + 40x_6 \ \wedge \\ & y_3^{(2)} \ge 800 + 140x_2 - 10 \ \wedge \\ & 0 \le x_1 \le 1 \ \wedge \cdots \wedge \ 0 \le x_6 \le 1 \ \wedge \ x_4 + x_6 \ge 2x_2). \end{aligned} \tag{4.12}$$

The polynomials occurring in (4.12) are also all linear, so we can employ a specialized QE algorithm [81] as well. This computation time is less than one second. Performing QE on (4.12), obtain an equivalent quantifier-free formula $\tau'(z, y_3^{(1)}, y_3^{(2)})$. To obtain the relation between $z$ and $y_3$, consider the following MOO problem:

$$\begin{cases} \text{Maximize} & z \text{ and} \\ \text{minimize} & y_3 = y_3^{(1)} \cdot y_3^{(2)} \\ \text{subject to} & \tau'(z, y_3^{(1)}, y_3^{(2)}) \quad . \end{cases} \tag{4.13}$$

Note that the MOO problem (4.13) is not equivalent to the original, although it has the same Pareto optimal front. From the numerical results obtain a plotting result in the $y_3$-$z$ plane via $y_3 = y_3^{(1)} \cdot y_3^{(2)}$. This combined method of QE and a numerical method is quite efficient and effective compared with conventional methods such as a GA-based method because one can reduce the number of decision variables optimized by a numerical method. This reduces the number of repetitions of numerical yield-rate evaluations required to obtain a comparable result.

A numerical optimization tool usually requires the ranges of the decision variables to influence computational efficiency. Fortunately, from (4.10) it is easy to obtain the range of $g_3^{(1)}(\boldsymbol{x})$ and $g_3^{(2)}(\boldsymbol{x})$. In addition, we have the maximal value $z_{\max}$ of $z$ from Problem 1. There is no minimal value of $z$ from the QE problem (4.12). However, we can assume that the minimal value of $z$ is zero because we are interested only in a high yield rate.

Now solve the following MOO problem with a numerical method as our symbolic-numeric approach:

$$\begin{cases} \text{Maximize} & z \text{ and} \\ \text{minimize} & y_3 = y_3^{(1)} \cdot y_3^{(2)} \\ \text{subject to} & \tau'(z, y_3^{(1)}, y_3^{(2)}) \quad , \\ & 1420 \leq y_3^{(1)} \leq 1920,\ 800 \leq y_3^{(2)} \leq 930,\ 0 \leq z \leq z_{\max} \quad . \end{cases} \tag{4.14}$$

Now we compare our approach with conventional numerical methods. The conventional numerical methods apply for the original optimization problem (4.10) directly. Figures 4.7 and 4.8 show the results for the problems (4.10) and (4.14) obtained by a PSO-based method after 2000 evaluations. We can say that our symbolic-numeric approach efficiently produces more optimal solutions than the conventional method.

Next we compare our approach with GA as another numerical method. Figures 4.10 and 4.11 show the results for the problems (4.10) and (4.14) obtained by a GA-based method after 2000 evaluations, respectively. We can say that in this problem the GA-based method works more effectively than the PSO-based method.

The result of the GA-based method shows that the samples obtained by a GA-based method crowd around the boundary, and the boundary is clearer than that of our symbolic-numeric method. It looks like the GA-based method is closer to optimal than our symbolic-numeric method. However, our symbolic-numeric method obtains a better approximation of the Pareto optimal front. Figure 4.12 shows plots of the non-dominated solution of the GA-based method (+) and our symbolic-numeric method (□).

In general, we are not able to know clearly whether the Pareto optimal front is obtained by a numerical method. However, in many cases fewer decision variables produce a more precise Pareto optimal front. Our method improves the precision of the Pareto optimal front under the same number of evaluations. In other words, our method reduces the number of evaluations required to obtain a comparable result.

One can eliminate one more decision variable for greater numerical efficiency. Since we have the result of a QE algorithm, it is easy to obtain the maximal value of the yield rate by using a bisection method. Obviously, from the QE problem (4.12) the following formulae must hold:

$$\begin{array}{ll} \forall y_3^{(1)} \forall y_3^{(2)} \forall z \forall z' & (z' \le z \wedge \tau'(z, y_3^{(1)}, y_3^{(2)}) \to \tau'(z', y_3^{(1)}, y_3^{(2)})). \\ \forall y_3^{(1)} \forall y_3^{(2)} \forall z \forall z' & (z' \ge z \wedge \neg\tau'(z, y_3^{(1)}, y_3^{(2)}) \to \neg\tau'(z', y_3^{(1)}, y_3^{(2)})). \end{array}$$

From this property we find the maximal value of $z$ for all $y_3^{(1)}$ and $y_3^{(2)}$ by using the bisection method.

We can reformulate (4.14) with the bisection method as follows:

$$\left\{ \begin{array}{ll} \text{Maximize} & z \text{ and} \\ \text{minimize} & y_3 = y_3^{(1)} \cdot y_3^{(2)} \\ \text{subject to} & z \equiv \textsc{Bisection}((y_3^{(1)}, y_3^{(2)}), [0, z_{\mathrm{m}ax}], \tau'), \\ & 1420 \le y_3^{(1)} \le 1920,\ 800 \le y_3^{(2)} \le 930 \ . \end{array} \right. \tag{4.15}$$

Figures 4.9 and 4.13 show the results obtained by our method with the bisection approach after 2000 evaluations. Since the number of decision variables is two, we can see a significant effect of the bisection approach for both numerical methods.

There is another advantage of our symbolic-numeric method. A QE algorithm for the formula (4.12) yields a feasible region $\tau'(z, y_3^{(1)}, y_3^{(2)})$. We used $\tau'(z, y_3^{(1)}, y_3^{(2)})$ as a constraint to form the formula (4.13) before switching the numerical part of the

**Algorithm 8** BISECTION($(x_1, \dots, x_k)$, $[z_l, z_u]$, $\phi(z, x_1, \dots, x_k)$)

**Input:** a vector of variables $(x_1, \dots, x_k)$, an interval $[z_l, z_u]$ and a semialgebraic set $\phi(z, x_1, \dots, x_k)$.

**Output:** maximal value of $z$ for $(x_1, \dots, x_k)$

```
$l \leftarrow z_l$
$u \leftarrow z_u$
loop
  $m \leftarrow (l + u)/2$
  if $\phi(m, x_1, \dots, x_k)$ then
    $l \leftarrow m$
  else
    $u \leftarrow m$
  end if
end loop
return $l$
```

method. However, the region specified by $\tau'(z, y_3^{(1)}, y_3^{(2)})$ is $(y_3^{(1)}, y_3^{(2)}, z)$ where the cell with width $y_3^{(1)}$ and length $y_3^{(2)}$ has the yield rate not less than $z$. A designer in our company requires a yield rate greater than 9/2. By substituting 9/2 for $z$ in $\tau'(z, y_3^{(1)}, y_3^{(2)})$, we obtain the relation between the width and the length of the cell with

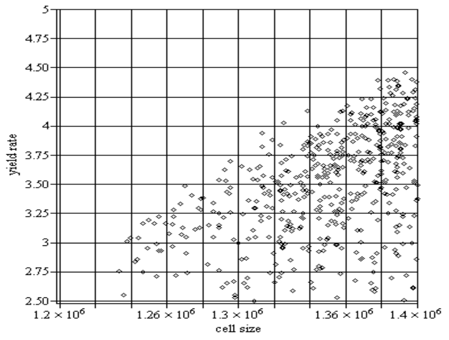

Figure 4.7: PSO-based method (2000 samples)

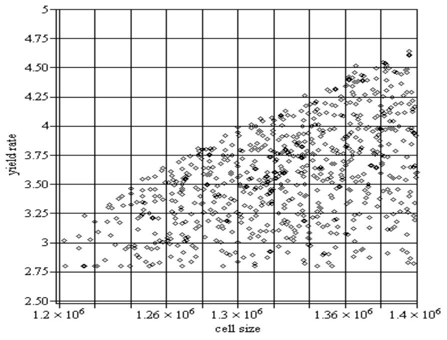

Figure 4.8: QE + PSO (2000 samples)

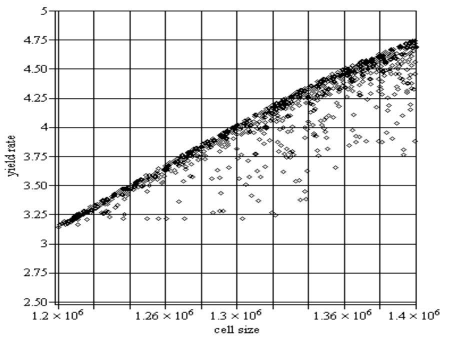

Figure 4.9: QE + PSO + bisection (2000 samples)

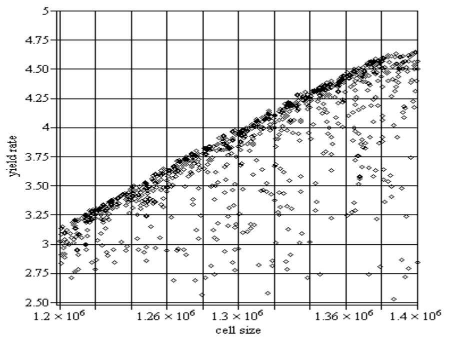

Figure 4.10: GA-based method (2000 samples)

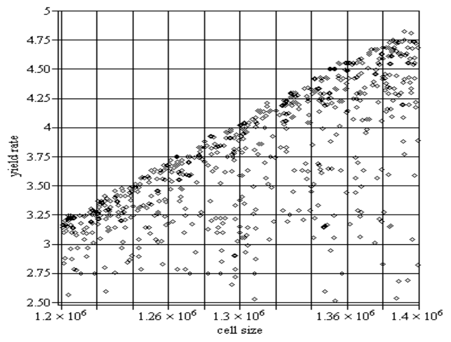

Figure 4.11: QE + GA (2000 samples)

a yield rate of 9/2 or higher, which is nearly optimal as indicated in Figure 4.15:

$$\begin{aligned}
&\tau'(9/2, y_3^{(1)}, y_3^{(2)}) = \\
&892595475289082211481274749 57y_3^{(1)} \geq 1431673894859616933462590161 89475 \ \wedge \\
&427475274237024206508954304121y_3^{(2)} \geq 347697181600191748755206249897360 \ \wedge \\
&119341432828219713253880415952y_3^{(1)}136269149560979337783764059 25y_3^{(2)} \\
&-2032073530174034389025178657 49215 \geq 0 \ \wedge \\
&835390029797537992777162911664y_3^{(1)}955750215209230487003872425725y_3^{(2)} \\
&-216236262816605272979512938 2699505 \geq 0 \ \wedge \\
&668312023838030394221730329 3312y_3^{(1)}106868818559256051627238576030 25y_3^{(2)} \\
&-198532403380999013023830527 47265040 \geq 0.
\end{aligned}$$

On the other hand, if a numerical method is directly applied one would form an op-

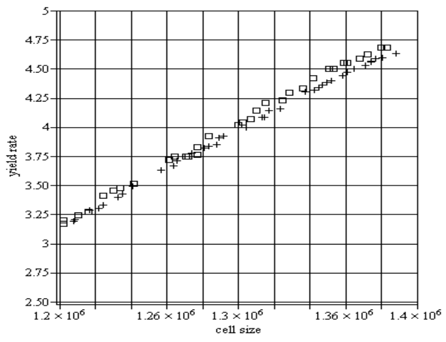

Figure 4.12: Non-dominated solution of GA (+) and QE + GA (□)

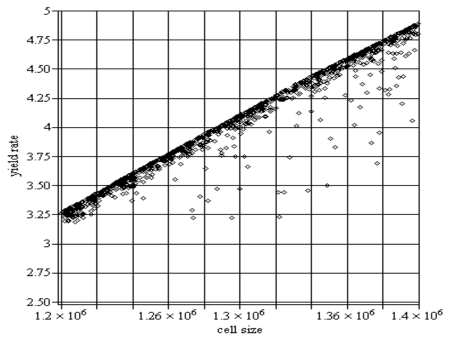

Figure 4.13: QE + GA + bisection (2000 samples)

timization problem (4.16). Figure 4.14 shows the result for (4.16) with a GA-based method after 5000 evaluations. Figure 4.15 describes the relation between two sides of the SRAM cell much more adequately than does Figure 4.14. Our QE-based method clearly shows the exact Pareto optimal front, while the samples obtained by a GA method crowd around the point $(y_3^{(1)}, y_3^{(2)}) = (1630, 850)$. It is hard to determine the trade-off between the sides.

$$\begin{cases} \text{Minimize} & y_3^{(1)} \text{ and} \\ \text{minimize} & y_3^{(2)} \\ \text{subject to} & y_1 \equiv g_1(\boldsymbol{x}),\ y_2 \equiv g_2(\boldsymbol{x}),\ \min(y_1, y_2) \geq 9/2, \\ & y_3^{(1)} \equiv g_3^{(1)}(\boldsymbol{x}),\ y_3^{(2)} \equiv g_3^{(2)}(\boldsymbol{x}), \\ & 0 \leq x_1 \leq 1,\ \ldots,\ 0 \leq x_6 \leq 1,\ x_4 + x_6 \geq 2x_2. \end{cases} \tag{4.16}$$

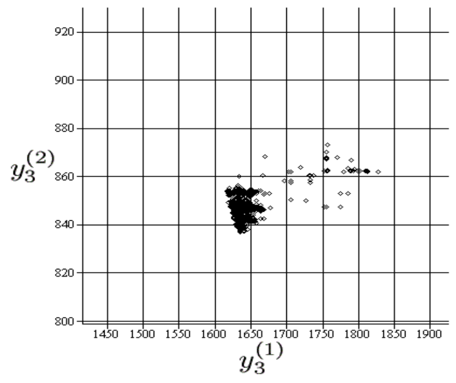

Figure 4.14: GA-based method (5000 samples)

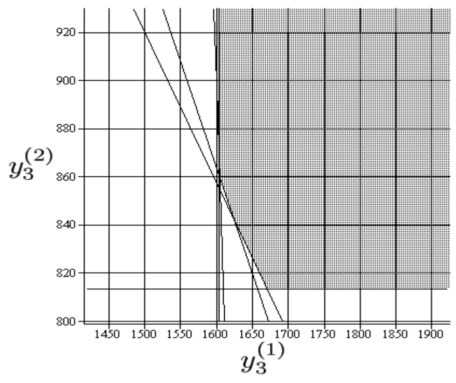

Figure 4.15: QE-based method $(\tau'(9/2, y_3^{(1)}, y_3^{(2)}))$

**Problem 4**

Our symbolic approach cannot deal with non-polynomial objective functions. Here we show that our symbolic-numeric approach can be applied to an MOO problem with a non-polynomial objective function using Structure 3.

Consider the following concrete MOO problem.

$$\begin{cases} \text{Maximize} & z \equiv \min(y_1, y_2) \text{ and} \\ \text{minimize} & y_4 \\ \text{subject to} & y_1 \equiv g_1(\boldsymbol{x}),\, y_2 \equiv g_2(\boldsymbol{x}),\, y_4 \equiv g_4(x_1, x_2), \\ & 0 \le x_1 \le 1,\, \ldots,\, 0 \le x_6 \le 1,\, x_4 + x_6 \ge 2x_2, \end{cases} \tag{4.17}$$

where

$$g_4(x_1, x_2) = \frac{5x_2 + 4}{x_1 + 3} e^{-2.8x_1 - 25x_2 - 18.4}.$$

The associated QE problem is given as

$$\begin{aligned} \exists x_1 \cdots \exists x_6 \quad & (z \le g_1(\boldsymbol{x}) \ \wedge \ z \le g_2(\boldsymbol{x}) \ \wedge \ y_4 = g_4(x_1, x_2) \ \wedge \\ & 0 \le x_1 \le 1 \ \wedge \cdots \wedge \ 0 \le x_6 \le 1 \ \wedge \ x_4 + x_6 \ge 2x_2). \end{aligned} \tag{4.18}$$

Because $g_4(x_1, x_2)$ is exponential, we cannot apply a QE algorithm to (4.18). Since $g_4$ depends only on two variables, we consider the following sub-problem:

$$\begin{aligned} \exists x_3 \cdots \exists x_6 \quad & (z \le g_1(\boldsymbol{x}) \ \wedge \ z \le g_2(\boldsymbol{x}) \ \wedge \\ & 0 \le x_1 \le 1 \ \wedge \cdots \wedge \ 0 \le x_6 \le 1 \ \wedge \ x_4 + x_6 \ge 2x_2). \end{aligned} \tag{4.19}$$

This QE problem consists of only polynomials, so we can apply a QE algorithm. Performing QE on (4.19), obtain an equivalent quantifier-free formula $\upsilon'(z, x_1, x_2)$. Reformulate (4.17) with $\upsilon'(z, x_1, x_2)$ as follows:

$$\begin{cases} \text{Maximize} & z \text{ and} \\ \text{minimize} & y_4 \\ \text{subject to} & \upsilon'(z, x_1, x_2),\, y_4 \equiv g_4(x_1, x_2), \\ & 0 \le x_1 \le 1,\, 0 \le x_2 \le 1, 0 \le z \le z_{max}, \end{cases} \tag{4.20}$$

where the range of $z$ is from (4.14). We note that the MOO problem (4.17) is not equivalent to the original one but has the same Pareto optimal front where $z \ge 0$.

Now we compare our method with a conventional numerical method with the GA tool, as in Problem 3. Figures 4.16 and 4.17 show the results for the optimization problems (4.17) and (4.20) obtained by the GA-based method after 5000 evaluations, respectively. The conventional numerical method applies GA for the original optimization problem (4.17) directly. On the other hand, our method applies GA for the optimization problem (4.20), which has only three decision variables. Our method improves the precision of the Pareto optimal front as in Problem 3.

Finally, we show the results of our symbolic-numeric approach with the bisection method. From the QE problem (4.19) the following formulae must hold:

$$\forall x_1 \forall x_2 \forall z \forall z' \quad (z' \leq z \wedge v'(z, x_1, x_2) \rightarrow v'(z', x_1, x_2)).$$
$$\forall x_1 \forall x_2 \forall z \forall z' \quad (z' \geq z \wedge \neg v'(z, x_1, x_2) \rightarrow \neg v'(z', x_1, x_2)).$$

Thus we can apply the bisection method. Reformulate (4.20) with the bisection method as follows:

$$\begin{cases} \text{Maximize} & z \text{ and} \\ \text{minimize} & y_4 \\ \text{subject to} & y_4 \equiv g_4(x_1, x_2),\ z \equiv \textsc{Bisection}((x_1, x_2), [0, z_{max}], v'), \\ & 0 \leq x_1 \leq 1,\ 0 \leq x_2 \leq 1. \end{cases} \tag{4.21}$$

Figure 4.18 shows the result for the optimization problem (4.21) obtained by the GA-based method after 2000 evaluations. Since the MOO problem (4.21) has only two variables, we can obtain a closer approximation of the Pareto optimal front than can other approaches.

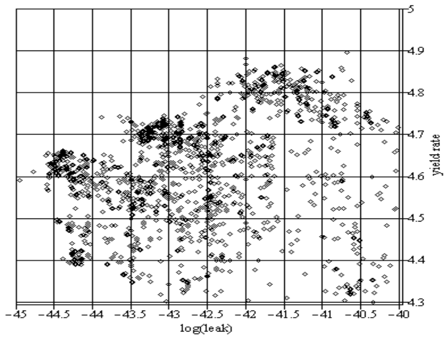

Figure 4.16: GA-based method (5000 samples)

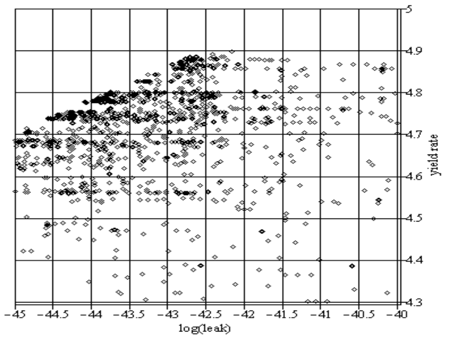

Figure 4.17: QE + GA (5000 samples)

## 4.6 Conclusions

We have proposed new symbolic-numeric optimization methods based on quantifier elimination combined with numerical computation for some important classes of opti-

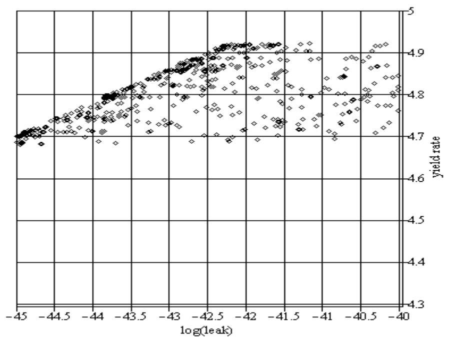

Figure 4.18: QE + GA + bisection (2000 samples)

mization problems in SRAM optimal design. The methods improve the total efficiency of the design process by reducing the number of numerical evaluations and also produce useful information such as more accurate relations among design parameters or objective functions.

The proposed methods have been employed in an industrial manufacturing process for a certain type of SRAM. Total design time has been dramatically reduced and, in one successful case, a design process which took about ten days is reduced to about nine days.

The contribution of this chapter is to show effective, concrete ways of using symbolic methods such as QE together with existing numerical methods. Future work includes exploring other applications in which we can apply the proposed approaches.

# Chapter 5

# Conclusion

Quantifier elimination (QE) is a powerful technique for solving problems over real closed fields. In this thesis, we have improved three types of QE approach and have successfully applied our algorithms in several applications.

First, we have improved cylindrical algebraic decomposition (CAD) for QE by using *quick tests*. We employ quick tests because it is expected that one can often detect an *unnecessary* procedure that may be skipped without violating the correctness of the results by running a simple test beforehand. By using quick tests, we have only constructed CAD sufficient to describe the solutions for a given first-order formula. We call such a CAD construction a *bounded CAD* construction. The effectiveness of our quick tests was verified using statistical data taken from many example problems.

Second, we have improved a special QE algorithm for sign definite conditions (SDCs) [45]. To improve the efficiency of the algorithm, *simplification* of output logical formulae is a critical issue. For this purpose, we have proposed two approaches: we have provided a necessary condition for the SDC problems to remove unnecessary sign conditions and have demonstrated Boolean function manipulation. We have also successfully applied our algorithm to a controller design procedure for a power supply unit [56].

Finally, we have proposed new symbolic-numeric optimization methods based on QE. We have constructed a first-order formula which expresses the feasible region for objective functions and eliminated decision variables by QE as much as possible. We have also applied a numerical optimization method to the smaller problem equivalent to the given problem. The methods improve the total efficiency of the design process by reducing the number of numerical evaluations and also produce useful information such as more accurate relations among design parameters or objective functions.

# Bibliography

[1] Espresso. `http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/`.

[2] Hirokazu Anai and Shinji Hara. Fixed-structure robust controller synthesis based on sign definite condition by a special quantifier elimination. In *Proceedings of American Control Conference, 2000*, volume 2, pages 1312–1316, 2000.

[3] Hirokazu Anai and Shinji Hara. A parameter space approach to fixed-order robust controller synthesis by quantifier elimination. *International Journal of Control*, 79(11):1321–1330, 2006.

[4] Hirokazu Anai, Shinji Hara, Masaaki Kanno, and Kazuhiro Yokoyama. Parametric polynomial spectral factorization using the sum of roots and its application to a control design problem. *Journal of Symbolic Computation*, 44(7):703–725, 2009.

[5] Hirokazu Anai and Pablo A. Parrilo. Convex quantifier elimination for semidefinite programming. In *Proceedings of the International Workshop on Computer Algebra in Scientific Computing (CASC 2003)*, pages 3–11, September 2003.

[6] Hirokazu Anai, Hitoshi Yanami, Shinji Hara, and Kei Sakabe. Fixed-structure robust controller synthesis based on symbolic-numeric computation: design algorithms with a CACSD toolbox. In *Proceedings of CCA/ISIC/CACSD 2004 (Taipei, Taiwan)*, volume 2, pages 1540–1545, September 2004.

[7] Hirokazu Anai and Kazuhiro Yokoyama. Cylindrical algebraic decomposition via numerical computation with validated symbolic reconstruction. In Andreas Dolzman, Andreas Seidl, and Thomas Sturm, editors, *Algorithmic Algebra and Logic*, pages 25–30, 2005.

[8] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[9] Robert King Brayton, Alberto L. Sangiovanni-Vincentelli, Curtis T. McMullen, and Gary D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1984.

[10] Christopher W. Brown. *Solution formula construction for truth invariant CAD's*. PhD thesis, University of Delaware Newark, 1999.

[11] Christopher W. Brown. Improved projection for CAD's of $\mathbb{R}^3$. In *Proceedings of the 2000 international symposium on Symbolic and algebraic computation*, ISSAC '00, pages 48–53, New York, NY, USA, 2000. ACM.

[12] Christopher W. Brown. Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 32(5):447–465, 2001.

[13] Christopher W. Brown. QEPCAD B: A program for computing with semi-algebraic sets using CADs. *SIGSAM BULLETIN*, 37:97–108, 2003.

[14] Christopher W. Brown and James H. Davenport. The complexity of quantifier elimination and cylindrical algebraic decomposition. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, ISSAC '07, pages 54–60, New York, NY, USA, 2007. ACM.

[15] Bob F. Caviness and Jeremy R Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition (Texts and Monographs in Symbolic Computation)*. Springer Vienna, softcover reprint of the original 1st ed. 1998 edition, April 1998.

[16] Changbo Chen and Marc Moreno Maza. An incremental algorithm for computing cylindrical algebraic decompositions. *CoRR*, abs/1210.5543, 2012.

[17] Changbo Chen, Marc Moreno Maza, Bican Xia, and Lu Yang. Computing cylindrical algebraic decomposition via triangular decomposition. *CoRR*, abs/0903.5221, 2009.

[18] M. D. Choi, T. Y. Lam, and Bruce Reznick. Even symmetric sextics. *Mathematische Zeitschrift*, 195(4):559–580, 1987.

[19] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages 2nd GI Confer-*

*ence Kaiserslautern, May 20-23, 1975*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183. Springer-Verlag, 1975.

[20] George E. Collins. *Quantifier elimination and cylindrical algebraic decomposition*, pages 8–23. In Caviness and Johnson [15], softcover reprint of the original 1st ed. 1998 edition, April 1998.

[21] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, 1991.

[22] George E. Collins, Jeremy R. Johnson, and Werner Krandick. Interval arithmetic in cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 34(2):145–157, 2002.

[23] Olivier Coudert, Jean Christophe Madre, and Henri Fraisse. A new viewpoint on two-level logic minimization. In *Proceedings of the Design Automation Conference*, pages 625–630, 1993.

[24] James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1/2):29–35, 1988.

[25] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, 2001.

[26] Jean Della Dora, Claire Dicrescenzo, and Dominique Duval. About a new method for computing in algebraic number fields. In *Research Contributions from the European Conference on Computer Algebra-Volume 2*, EUROCAL '85, pages 289–290, London, UK, 1985. Springer-Verlag.

[27] Andreas Dolzmann, Andreas Seidl, and Thomas Sturm. Efficient projection orders for CAD. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, ISSAC '04, pages 111–118, New York, NY, USA, 2004. ACM.

[28] Andreas Dolzmann and Thomas Sturm. REDLOG computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31:2–9, 1996.

[29] Andreas Dolzmann, Thomas Sturm, and Volker Weispfenning. Real quantifier elimination in practice. In *Algorithmic Algebra and Number Theory*, pages 221–247. Springer, 1998.

[30] Dominique Duval. Algebraic numbers: An example of dynamic evaluation. *Journal of Symbolic Computation*, 18(5):429–445, 1994.

[31] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Nagoya, Japan, 1995.

[32] Agoston E. Eiben and James E. Smith. *Introduction to Evolutionary Computing.* Springer-Verlag, 2003.

[33] David E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley, 1989.

[34] Laureano González-Vega. Applying quantifier elimination to the Birkhoff interpolation problem. *Journal of Symbolic Computation*, 22:83–103, July 1996.

[35] Laureano González-Vega. *A combinatorial algorithm solving some quantifier elimination problems*, pages 365–375. In Caviness and Johnson [15], softcover reprint of the original 1st ed. 1998 edition, April 1998.

[36] Laureano González-Vega, T Recio, H Lombardi, and M.-F Roy. *Sturm-Habicht sequences determinants and real roots of univariate polynomials*, pages 300–316. In Caviness, Bob F. and Johnson, Jeremy R [15], softcover reprint of the original 1st ed. 1998 edition, April 1998.

[37] Feng Guo, Mohab Safey El Din, and Lihong Zhi. Global optimization of polynomials using generalized critical values and sums of squares. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, ISSAC '10, pages 107–114, New York, NY, USA, 2010. ACM.

[38] Yacov Y. Haimes. Integrated system identification and optimization. In *Control and Dynamic Systems: Advances in Theory and Applications*, volume 10, pages 435–518. Academic Press, 1973.

[39] William R. Harris. Real even symmetric ternary forms. *Journal of Algebra*, 222(1):204 – 245, 1999.

[40] Hong Hong and Mohab Safey El Din. Variant quantifier elimination. *Journal of Symbolic Computation*, 47(7):883–901, 2012.

[41] Hoon Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proceedings of the international symposium on Symbolic and algebraic computation*, pages 261–264, New York, NY, USA, 1990. ACM.

[42] Hoon Hong. Quantifier elimination for formulas constrained by quadratic equations. In *Proceedings of the 1993 international symposium on Symbolic and algebraic computation*, ISSAC '93, pages 264–274, New York, NY, USA, 1993. ACM.

[43] Hoon Hong. An efficient method for analyzing the topology of plane real algebraic curves. In *Selected papers presented at the international IMACS symposium on Symbolic computation, new trends and developments*, pages 571–582, Amsterdam, The Netherlands, The Netherlands, 1996. Elsevier Science Publishers B. V.

[44] Hoon Hong. *Simple Solution Formula Construction in Cylindrical Algebraic Decomposition Based Quantifier Elimination*, pages 201–219. In Caviness and Johnson [15], softcover reprint of the original 1st ed. 1998 edition, April 1998.

[45] Noriko Hyodo, Myunghoon Hong, Hitoshi Yanami, Shinji Hara, and Hirokazu Anai. Solving and visualizing nonlinear parametric constraints in control based on quantifier elimination. *Applicable Algebra in Engineering, Communication and Computing*, 18(6):497–512, 2007.

[46] Hidenao Iwane, Hiroyuki Higuchi, and Hirokazu Anai. An effective implementation of a special quantifier elimination for a sign definite condition by logical formula simplification. In Vladimir P. Gerdt, Wolfram Koepf, Ernst W. Mayr, and Evgenii V. Vorozhtsov, editors, *CASC*, volume 8136 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2013.

[47] Hidenao Iwane, Akifumi Kira, and Hirokazu Anai. Construction of explicit optimal value functions by a symbolic-numeric cylindrical algebraic decomposition. In Vladimir P. Gerdt, Wolfram Koepf, Ernst W. Mayr, and Evgenii V. Vorozhtsov, editors, *CASC*, volume 6885 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2011.

[48] Hidenao Iwane, Hitoshi Yanami, and Hirokazu Anai. An effective implementation of a symbolic-numeric cylindrical algebraic decomposition for optimization prob-

lems. In *Proceedings of the 2011 International Workshop on Symbolic-Numeric Computation*, volume 1, pages 168–177, 2011.

[49] Hidenao Iwane, Hitoshi Yanami, Hirokazu Anai, and Kazuhiro Yokoyama. An effective implementation of a symbolic-numeric cylindrical algebraic decomposition for quantifier elimination. In *Proceedings of the 2009 International Workshop on Symbolic-Numeric Computation*, volume 1, pages 55–64, 2009.

[50] Hidenao Iwane, Hitoshi Yanami, Hirokazu Anai, and Kazuhiro Yokoyama. An effective implementation of symbolic–numeric cylindrical algebraic decomposition for quantifier elimination. *Theoretical Computer Science*, 479:43–69, 2013.

[51] Erich Kaltofen, Bin Li, Zhengfeng Yang, and Lihong Zhi. Exact certification of global optimality of approximate factorizations via rationalizing sums-of-squares with floating point scalars. In *Proceedings of the twenty-first international symposium on Symbolic and algebraic computation*, ISSAC '08, pages 155–164, New York, NY, USA, 2008. ACM.

[52] Rudolf Krawczyk. Newton-algorithmen zur bestimmung von nullstellen mit fehlerschranken. *Computing*, pages 187–201, 1969.

[53] Jouni Lampinen. *Multiobjective Nonlinear Pareto-Optimization.* Laboratory of Information Processing, 2000.

[54] Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[55] Rüdiger Loos and Volker Weispfenning. Applying linear quantifier elimination. *The Computer Journal*, 36(5):450–462, 1993.

[56] Yoshinobu Matsui, Hidenao Iwane, and Hirokazu Anai. Two controller design procedures using sdp and qe for a power supply unit. In *Development of Computer Algebra Research and Collaboration with Industry*, volume 49 of *COE Lecture Note*, pages 43–51, 2013.

[57] Scott McCallum. Solving polynomial strict inequalities using cylindrical algebraic decomposition. *The Computer Journal*, pages 432–438, 1993.

[58] Scott McCallum. An improved projection operator for cylindrical algebraic decomposition. In Caviness and Johnson [15], pages 242–268.

[59] Scott McCallum. On projection in CAD-based quantifier elimination with equational constraint. In *Proceedings of the 1999 international symposium on Symbolic and algebraic computation*, ISSAC '99, pages 145–149, New York, NY, USA, 1999. ACM.

[60] Scott McCallum. On propagation of equational constraints in CAD-based quantifier elimination. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, ISSAC '01, pages 223–231, New York, NY, USA, 2001. ACM.

[61] Patirick McGeer, Jagesh Sanghavi, Robert Brayton, and Alberto Sangiovanni Vincentelli. ESPRESSO-SIGNATURE: A new exact minimizer for logic functions. In *Proceedings of the Design Automation Conference*, pages 618–624, 1993.

[62] Kaisa Miettinen. *Nonlinear multiobjective optimization.* Kluwer Academic Publishers, Boston, 1999.

[63] Pablo A. Parrilo and Sanjay Lall. Semidefinite programming relaxations and algebraic optimization in control, 2006. Mini-course on Polynomial Equations and Inequalities I and II.

[64] Pablo A. Parrilo and Bernd Sturmfels. Minimizing polynomial functions. In *Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science*, volume 60, pages 83–99. ACM, 2001.

[65] Efstratios N. Pistikopoulos, Michael C. Georgiadis, and Vivek Dua. *Multiparametric programming: theory, algorithms, and applications*, volume 1 of *Multi-Parametric Programming.* Wiley-VCH, 2007.

[66] Stefan Ratschan. Approximate quantified constraint solving by cylindrical box decomposition. *Reliable Computing*, 8(1):21–42, 2002.

[67] Siegfried M. Rump. Algebraic computation, numerical computation and verified inclusions. In Rainer Janßen, editor, *Trends in Computer Algebra*, volume 296 of *Lecture Notes in Computer Science*, pages 177–197. Springer Berlin / Heidelberg, 1988.

[68] Siegfried M. Rump. *Computer-Assisted Proofs and Self-Validating Methods*, volume 18, chapter 10, pages 195–240. Society for Industrial and Applied Mathematics, Philadephia, PA, 2005.

[69] Mohab Safey El Din. Computing the global optimum of a multivariate polynomial over the reals. In *Proceedings of the twenty-first international symposium on Symbolic and algebraic computation*, ISSAC '08, pages 71–78, New York, NY, USA, 2008. ACM.

[70] Kumara Sastry. Single and multiobjective genetic algorithm toolbox in C++. `http://www.kumarasastry.com/2007/06/11/single-and-multiobjective-genetic-algorithm-toolbox-in-c/`.

[71] Jason Schattman. Portfolio optimization under nonconvex transaction costs with the global optimization toolbox. `http://www.maplesoft.com/applications/view.aspx?SID=1401&view=html`.

[72] Markus Schweighofer. Global optimization of polynomials using gradient tentacles and sums of squares. *SIAM Journal on Optimization*, 17:920–942, September 2006.

[73] Adam W. Strzeboński. A real polynomial decision algorithm using arbitrary-precision floating point arithmetic. *Reliable Computing*, 5(3):337–346, 1999.

[74] Adam W. Strzeboński. Solving systems of strict polynomial inequalities. *Journal of Symbolic Computation*, 29:471–480, March 2000.

[75] Adam W. Strzeboński. Cylindrical algebraic decomposition using validated numerics. *Journal of Symbolic Computation*, 41(9):1021–1038, 2006.

[76] Thomas Sturm. New domains for applied quantifier elimination. In Victor G. Ganzha, Ernst W. Mayr, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, volume 4194 of *Lecture Notes in Computer Science*, pages 295–301. Springer, 2006.

[77] Thomas Sturm and Andreas Weber. Investigating generic methods to solve hopf bifurcation problems in algebraic biology. In Katsuhisa Horimoto, Georg Regensburger, Markus Rosenkranz, and Hiroshi Yoshida, editors, *Algebraic Biology*, volume 5147 of *Lecture Notes in Computer Science*, chapter 15, pages 200–215. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2008.

[78] Thomas Sturm and Volker Weispfenning. Rounding and blending of solids by a real elimination method. In *Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling, and Applied Mathematics (IMACS 97)*, pages 727–732, 1997.

[79] Alfred Tarski. *A decision method for elementary algebra and geometry.* University of California Press, 2nd edition, 1952.

[80] Vlad Timofte. On the positivity of symmetric polynomial functions. part ii: Lattice general results and positivity criteria for degrees 4 and 5. *Journal of Mathematical Analysis and Applications*, 304(2):652 – 667, 2005.

[81] Volker Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5:3–27, Febrary 1988.

[82] Volker Weispfenning. Quantifier elimination for real algebra - the quadratic case and beyond. *Applicable Algebra in Engineering, Communication and Computing*, 8:85–101, 1993.

[83] Volker Weispfenning. Simulation and optimization by quantifier elimination. *Journal of Symbolic Computation*, 24:189–208, August 1997.

[84] Benjamin Wilson, David Cappelleri, Timothy W. Simpson, and Mary Frecker. Efficient Pareto frontier exploration using surrogate approximations. *Optimization and Engineering*, 2:31–50, 2001.

[85] David J. Wilson, Russell J. Bradford, and James H. Davenport. Speeding up cylindrical algebraic decomposition by Gröbner bases. *CoRR*, abs/1205.6285, 2012.

[86] Franz Winkler. *Polynomial Algorithms in Computer Algebra.* Texts and Monographs in Symbolic Computation. Springer, 1996.

[87] Hitoshi Yanami. Multi-objective design based on symbolic computation and its application to hard disk slider design. *Journal of Math-for-Industry*, 1:149–156, 2009.

[88] Hitoshi Yanami and Hirokazu Anai. The Maple package SyNRAC and its application to robust control design. *Future Generation Computer Systems*, 23(5):721–726, 2007.

[89] Lu Yang and Bican Xia. Real solution classification for parametric semi-algebraic systems. In Andreas Dolzmann, Andreas Seidl, and Thomas Sturm, editors, *Algorithmic Algebra and Logic*, pages 281–289. Books on Demand, 2005.

[90] Lotfi Zadeh. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, 8:59–60, 1963.