# Zero Determination of Algebraic Numbers using Approximate Computation and its Application to Algorithms in Computer Algebra

Hiroshi Sekigawa
NTT Communication Science Laboratories,
Nippon Telegraph and Telephone Corporation
3-1, Morinosato Wakamiya, Atsugi-shi,
Kanagawa, 243-0198 Japan
E-mail: sekigawa@theory.brl.ntt.co.jp

## Abstract

The aim of this paper is to propose an alternative method for computation with algebraic numbers, combining the interval arithmetic, a kind of approximate numerical computation, and the estimation of a certain measure of algebraic numbers called the Mahler measure, and to show that this enables rigorous decision whether or not the result of a successive application of ring operations (namely addition, subtraction and multiplication) to a given set of algebraic numbers $\alpha_1$, $\alpha_2$, $\ldots$, $\alpha_n$ reduces to zero. For the purpose of numerical computation, we regard $\overline{\mathbb{Q}}$ as embedded into $\mathbb{C}$, and by the value of an algebraic number, we refer to its complex number value under this embedding. We also assume that each of the input numbers $\alpha_i$ is given in terms of a polynomial $f_i(x) \in \mathbb{Z}[x]$ vanishing at $\alpha_i$ and an interval isolating $\alpha_i$ from the other roots of $f_i(x)$.

We sharpen inequalities on the Mahler measure after performing ring operations among two algebraic numbers and we propose two methods for applying interval computations and the Mahler measure computations. One method computes both intervals and the Mahler measures simultaneously. The other method utilizes a history of computation to estimate the Mahler measures only when they are required. We also report some experimental results of applying the methods to construct two-dimensional convex hulls.

# Contents

# List of Figures

# List of Tables

## 1 Introduction

The aim of this paper is to propose an alternative method for computation with algebraic numbers, combining the interval arithmetic and the estimation of a certain measure of algebraic numbers called the Mahler measure (see Definition 5), and to show that this enables rigorous decision whether or not the result $\alpha$ of a successive application of ring operations (namely addition, subtraction and multiplication) to a given set of algebraic numbers $\alpha_1$, $\alpha_2$, $\dots$, $\alpha_n$ reduces to zero.

In computation with algebraic numbers, exact arithmetic is possible (see Discussion at the end of this paper) but can be costly. In such cases, it is natural to attempt approximate numerical computation, which is usually much more efficient, and to try to extract useful information. In this paper we concentrate on using interval arithmetic (see 2.1.1), which is a form of numerical computation, increasingly in wide use in situations where a more rigorous treatment of computational errors than the ordinary single-value floating-point arithmetic is desired. For the purpose of numerical computation, we regard $\overline{\mathbb{Q}}$ as embedded into $\mathbb{C}$, and by the value of an algebraic number, we refer to its complex number value under this embedding. We also assume that each of the input numbers $\alpha_i$ is given in terms of a polynomial $f_i(x) \in \mathbb{Z}[x]$ vanishing at $\alpha_i$ and an interval isolating $\alpha_i$ from the other roots of $f_i(x)$.

However, in general, the output from an algorithm based on exact arithmetic, naively applied to an approximate input, may not be even close to the true output; and even if we let the input approach the true value indefinitely, the output may not approach the true output. This property is called instability. For simplicity, let us concentrate on an algorithm consisting of ring operations and branchings on equality conditions only. Although the ring operations are continuous, the equality conditions do have incontinuity at the very point where the two sides are equal, which causes the succeeding computation to follow completely different paths. Thus instability arises in the purest form if the problem itself is to decide whether or not zero is the result of a certain sequence of computation. Such a problem is called a zero determination.

Shirayanagi proposed a method for stabilizing Buchberger's algorithm, which computes the Gröbner basis of an ideal of a ring [29], [30]. The method uses interval computation endowed with "zero rewriting." Zero rewriting is the rule that prescribes to replace any interval containing zero by a single point zero whenever such an interval appears in the course of computation. The underlying ideas of this method were generalized by Shirayanagi and Sweedler as a theory of stabilizing algebraic algorithms [31]; an algorithm thus “stabilized” acquires the property that, if we let the input interval approach the true input value, then the output approaches what should be the result of exact computation applied to the limit input value.

There is a delicate point with this method. For simplicity, suppose the computation proceeds up to a certain point without zero rewriting. If the interval $I$ obtained at this point does not contain zero, then the corresponding true value $\alpha$, being contained in $I$, cannot be zero. On the other hand, if $I \ni 0$, one cannot confirm that $\alpha = 0$ solely from this computation, since this can also happen if $\alpha \neq 0$ but the precision is insufficient. Since the true value is not known, there is no way in general to know how far one needs to raise the precision in order to separate it from zero if $\alpha \neq 0$; in case $\alpha = 0$, this situation of uncertainty persists how far one raises the precision.

Our point is that, if $\alpha$ is the result of applying a succession of ring operations on given algebraic numbers $\alpha_1$, $\alpha_2$, $\ldots$, $\alpha_n$ (see Definition 1 for a precise formulation), one can find a ball $B$ around the origin with positive radius $\varepsilon$ such that $\alpha$ can fall inside $B$ only if it is zero. Such $\varepsilon$ is called a separation bound for $\alpha$ (see Definition 4). If one knows such a bound $\varepsilon$ together with an interval $I$ assured to contain $\alpha$, one can (a) decide that $\alpha \neq 0$ if $0 \notin I$; or (b) decide that $\alpha = 0$ if $0 \in I \subset B$; or otherwise (c) know a precision of $\alpha$ required for decision, and by iterating the computation with increased precisions (possibly more than once), one can reach (a) or (b) (see Theorem 1).

Such $\varepsilon$ can be found because, in principle, a polynomial $f(x) \in \mathbb{Z}[x]$ vanishing at $\alpha$ can be derived from the given information on the $\alpha_i$. However, computing $f$ can be costly. In practice, a general framework for finding such $\varepsilon$ without computing $f$ can be constructed by choosing a family $X$ of conditions (predicates) defined on $\overline{\mathbb{Q}}$ satisfying the following requirements. For $X$ we propose to choose all assertions of upper estimates of the degree and the Mahler measure, namely all conditions of the form “$\deg \alpha \leq d$ and $M(\alpha) \leq A$,” $d \in \mathbb{N}$ and $A \geq 1$, where $\deg \alpha = [\mathbb{Q}(\alpha) : \mathbb{Q}]$ and $M(\alpha)$ is the Mahler measure of $\alpha$. The requirements are as follows: (1) for each $\alpha_i \in \overline{\mathbb{Q}}$ given as input, a predicate $\xi \in X$ satisfied by $\alpha_i$ can be easily found; (2) from two predicates $\xi$, $\eta \in X$, a third predicate $\zeta \in X$ such that “if $\alpha$, $\beta \in \overline{\mathbb{Q}}$ satisfy $\xi$, $\eta$ respectively, then $\alpha * \beta$ satisfies $\zeta$” can be easily found ($* \in \{+, -, \times\}$), and (3) from a predicate $\xi \in X$, a bound $\varepsilon > 0$ such that “if $\alpha \in \overline{\mathbb{Q}}$ satisfies $\xi$, then either $\alpha = 0$ or $|\alpha| \geq \varepsilon$” can be easily found (see Theorem 2). Although there are known inequalities showing that our choice of $X$ fulfills the requirements (1)–(3) (see Proposition 1), we further improve the inequalities for the recurrent step (2) in two ways (see Propositions 4 and 5; in fact, the latter also uses the approximate values of $\alpha$, $\beta$ to derive $\zeta$). Still the amount of computation for (1)–(3) is small. By pursuing these conditions using (1) and (2) along with the computation, and finally using (3) just before the equality test, a separation bound is obtained. Note that, if $\alpha$ is real, then its signature is also determined along with the decision that $\alpha \neq 0$.

Also, in a sense similar to the stabilization of algorithms by incorporating the zero rewriting (see [31]), our method allows systematic conversion of an algorithm consisting of ring operations and branchings on equality conditions (and on signature conditions if the numbers are real), written in terms of exact arithmetic, into one for algebraic numbers based on interval arithmetic (see Section 3.2).

This paper is based on [26], [27], [28]. Johnson proposed a method using interval arithmetic for real algebraic number computation [13]. However, his method resorts to exact arithmetic when an interval contains zero. After publication of [28], some works in the same direction as ours have been done in the context of computational geometry; see [3], [20] and [17] for example.

The construction of the paper is as follows. In Section 2, after describing our idea and the principle of zero determination that uses intervals and additional information on separation bounds, we review the definition and properties of the Mahler measure which we use in finding as separation bounds. Also in this section, we refine the key inequalities on the Mahler measure to be used in step (2). In Section 3, we discuss some methods of the principle and show some examples. Finally, we summarize our results and describe future directions.

## 2 Zero Determination of Algebraic Numbers using Approximate Computation

### 2.1 Principle

#### 2.1.1 Idea

In this section, we formulate in general terms our principle of zero determination for the result of a successive application of ring operations on given algebraic numbers.

To clarify the concept of "the result of a successive application of ring operations on given algebraic numbers," the terminology of straight-line programs needs to be defined. A straight-line program here is a kind of a division-free non-scalar straight-line program; see for instance [14], [10].

**Definition 1 (straight-line program)** Let $P$ be a polynomial belonging to $\mathbb{Z}[x_1, x_2, \ldots, x_n]$. A sequence $(P_1, P_2, \ldots, P_r)$ is called a straight-line program for $P$ if the following conditions are satisfied:

1. For each $P_i$ ($1 \leq i \leq r$), either
$$P_i = x_j, \qquad 1 \leq j \leq n,$$
or
$$P_i = P_j * P_k, \qquad 1 \leq j, k < i \text{ and } * \in \{+, -, \times\}.$$
Here, $P_j * P_k$ is just a sequence of three symbols "$P_j$", "$+$" (or "$-$" or "$\times$") and "$P_k$".

2. If we interpret $* \in \{+, -, \times\}$ as a ring operation in $\mathbb{Z}[x_1, x_2, \ldots, x_n]$ and substitute each $P_i$ into either $x_j$ or $P_j * P_k$ recursively, then $P = P_r$.

We say that an algebraic number $\alpha$ is represented in algebraic numbers $\alpha_1$, $\alpha_2$, $\ldots$, $\alpha_n$ by a straight-line program when $\alpha = P(\alpha_1, \alpha_2, \ldots, \alpha_n)$ for a polynomial $P \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$ and $P$ is represented by a straight-line program.

That is, by representing an algebraic number $\alpha$ by a straight-line program in algebraic numbers $\alpha_i$'s, we specify the order of ring operations among $\alpha_i$'s for obtaining $\alpha$.

There are several different notions of intervals often used in interval arithmetic, such as the usual bounded closed intervals in $\mathbb{R}$ and the rectangular and the circular intervals in $\mathbb{C}$. Also, the bounded closed interval arithmetic can be constructed on different arithmetic on the endpoints, such as the exact arithmetic on rational numbers or the floating-point arithmetic, and in the case of floating-point arithmetic there are also different choices for the base of numeration such as decimal or binary. There are a similar variety of choices for the intervals in $\mathbb{C}$ (for a general reference see [1], for example). In order to make the discussion independent of the specific choice of the intervals, we set up the following definition.

**Definition 2 (interval arithmetic system)** Let $K$ be either $\mathbb{R}$ or $\mathbb{C}$.

A family $\mathcal{I}$ of connected compact subsets of $K$, given together with a way to encode them in finite sequences in a finite alphabet, will be called an "interval system" in $K$ if, for every $\alpha \in K$, there is a decreasing sequence $I_1 \supset I_2 \supset \cdots$ of elements of $\mathcal{I}$ containing $\alpha$ such that, for every $\varepsilon > 0$, all terms $I_\mu$ starting from a certain index $\mu_0$, depending on $\varepsilon$, satisfy $I_\mu \subset \{\, x \in K \mid |x - \alpha| < \varepsilon \,\}$. Such a sequence $\{I_\mu\}$ will be called an "approximate interval sequence" for $\alpha$ in $\mathcal{I}$, and we also write $\{I_\mu\} \to \alpha$ if this holds. We may also say that $I_\mu$ has precision $\mu$ in this context. If $\mathcal{I}$ is fixed, an element of $\mathcal{I}$ will be simply called an "interval."

By an "interval arithmetic system" we mean a quintuple $\boldsymbol{I} = (K, \mathcal{I}, +, -, \times)$ in which $\mathcal{I}$ is an interval system in $K$, and each of $* \in \{+, -, \times\}$ is a map $\mathcal{I} \times \mathcal{I} \to \mathcal{I}$, computable in terms of the given encoding, satisfying:

1. For any $I$, $J \in \mathcal{I}$, $I * J$ contains $\{\, a * b \mid a \in I, b \in J \,\}$.
2. If $I$, $I'$, $J$, $J'$ satisfy $I \supset I'$ and $J \supset J'$, then $I * J \supset I' * J'$.
3. If $\{I_\mu\} \to \alpha$ and $\{J_\mu\} \to \beta$, where $\alpha$, $\beta \in K$, then $\{I_\mu * J_\mu\} \to \alpha * \beta$.

In 1 and 3, the operation $*$ applied to the elements of $K$ denotes the usual operation $+$, $-$ and $\times$ in $K$.

Note that these are minimal requirements focused on the condition 3. In practice, as in the examples shown in Example 1 below, it is desirable to make $I * J$ as close to $\{\, a * b \mid a \in I, b \in J \,\}$ as possible in order to attain efficiency.

The following definition is a concept of evaluating a straight-line program in an interval arithmetic system.

**Definition 3** Let $\boldsymbol{P} = (P_1, P_2, \ldots, P_r)$ be a straight-line program for a polynomial $P \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$. Fix an interval arithmetic system $(K, \mathcal{I}, +, -, \times)$ and let $I_1$, $I_2$, $\ldots$, $I_n$ be intervals in $\mathcal{I}$. We say that an interval $I \in \mathcal{I}$ is the evaluation of the straight-line program $\boldsymbol{P}$ for intervals $I_1$, $I_2$, $\ldots$, $I_n$, if $I$ is obtained by interval arithmetic among intervals $I_1$, $I_2$, $\ldots$, $I_n$ with the order of operations as represented by the straight-line program $\boldsymbol{P}$.

Fix an interval arithmetic system $(K, \mathcal{I}, +, -, \times)$. Let $\alpha_1$, $\alpha_2$, $\ldots$, $\alpha_n \in K$ be given algebraic numbers and $\{I_{1,\mu}\}$, $\{I_{2,\mu}\}$, $\ldots$, $\{I_{n,\mu}\}$ be approximate interval sequences for $\alpha_1$, $\alpha_2$, $\ldots$, $\alpha_n$, respectively. Then, for a straight-line program $\boldsymbol{P}$ for a polynomial $P \in \mathbb{Z}[x_1, x_2, \ldots, x_n]$, the interval sequence $\{I_\mu\}$, where $I_\mu$ is the evaluation of the straight-line program $\boldsymbol{P}$ for intervals $I_{1,\mu}$, $I_{2,\mu}$, $\ldots$, $I_{n,\mu}$, is an approximate interval sequence for $\alpha = P(\alpha_1, \alpha_2, \ldots, \alpha_n)$.

**Example 1** We show two examples for the case $K = \mathbb{R}$, namely the one based on exact rational arithmetic and the one based on floating-point arithmetic.

By the interval arithmetic system based on exact rational arithmetic, we mean a system $(\mathbb{R}, \mathcal{I}, +, -, \times)$ where $\mathcal{I} = \{\, [a, b] \mid a, b \in \mathbb{Q}, a \le b \,\}$ and the operations $+$, $-$, $\times$ are defined by the traditional interval arithmetic operations, namely if $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$, then

$$I_1 + I_2 = [a_1 + a_2, b_1 + b_2], \qquad I_1 - I_2 = [a_1 - b_2, b_1 - a_2],$$

$$I_1 \times I_2 = [\min E, \max E], \text{ where } E = \{a_1a_2, a_1b_2, b_1a_2, b_1b_2\}.$$

An example of approximate interval sequence $\{I_\mu\}$ for $\alpha$ can be given by putting $I_\mu = [\alpha_\mu^-, \alpha_\mu^+]$ where $\alpha_\mu^-$ and $\alpha_\mu^+$ are the smaller and the larger of the $\mu$th and the $(\mu+1)$th truncations of the continued fraction expansion of $\alpha$.

For the interval arithmetic system based on floating-point arithmetic, the above formulation is actually an oversimplification. For instance, consider the floating-point arithmetic with base 10, and let $\mathcal{D}$ be the set of all finite decimals (including the integers). We put $\mathcal{I} = \{[a, b] \mid a, b \in \mathcal{D}, a \le b\}$, but each of the operations $+$, $-$, $\times$ should actually be regarded a family of operations $\{*_\mu\}$ parameterized by the computational precision $\mu$. Note that $\mathcal{D}$ has a filtration by the subsets $\mathcal{D}_\mu$ consisting of 0 and the decimals having at most $\mu$ significant digits, and $\mathcal{I}$ by the subsets $\mathcal{I}_\mu$ consisting of the intervals whose endpoints are in $\mathcal{D}_\mu$. For each $\mu \in \mathbb{N}$ and $* \in \{+, -, \times\}$, the operation $*_\mu\colon \mathcal{I}_\mu \times \mathcal{I}_\mu \to \mathcal{I}_\mu$ takes the pair $(I_1, I_2) \in \mathcal{I}_\mu \times \mathcal{I}_\mu$ to the smallest interval in $\mathcal{I}_\mu$ containing $\{\, x * y \mid x \in I_1,\ y \in I_2 \,\}$. This is well defined since the only accumulation point $\mathcal{D}_\mu$ is 0, which lies in $\mathcal{D}_\mu$.

However, specifying the computational precision with the operation would make the notation rather cumbersome. Instead, we stick to the previous notation, and we abuse it by making the following convention in the case of floating-point interval arithmetic. If we say that $\{I_\mu\}$ is an approximate interval sequence for $\alpha$, we always imply that $I_\mu \in \mathcal{I}_\mu$ for every $\mu$. If $\{I_\mu\} \to \alpha$ and $\{J_\mu\} \to \beta$, then applying an operation $* \in \{+, -, \times\}$ to $I_\mu$ and $J_\mu$ will always mean applying the operation $*_\mu$ to $I_\mu$ and $J_\mu$. Note that, with this understanding, the "continuity" condition of the definition of the interval arithmetic system is fulfilled; namely if $\{I_\mu\} \to \alpha$ and $\{J_\mu\} \to \beta$, then $\{I_\mu *_\mu J_\mu\} \to \alpha * \beta$ for $* \in \{+, -, \times\}$. Thus, if $\alpha$ is represented by a straight-line program $\boldsymbol{P} = (P_1, P_2, \dots, P_r)$ in terms of $\alpha_1$, $\alpha_2$, …, $\alpha_n$, and if $\{I_{i,\mu}\}$ is an approximate interval sequence for each $\alpha_i$, then to evaluate $\boldsymbol{P}$ for $I_{1,\mu}$, $I_{2,\mu}$, …, $I_{n,\mu}$ is to iterate the following operations for $j = 1$, 2, …, $r$ and take the interval assigned to $P_r$ at the final step: (1) if $P_j = x_k$, then assign $I_{k,\mu}$ to $P_j$, or (2) if $P_j = P_k * P_l$, then apply $*_\mu$ to the two intervals assigned to $P_k$ and $P_l$, and assign the result to $P_j$; if we denote the final interval by $I_\mu$, then $\{I_\mu\}$ is an approximate interval sequence for $\alpha$.

For an algebraic number $\alpha$ represented by a straight-line program, we can accurately determine that $\alpha$ is not equal to zero solely by numeric computation (e.g., by using traditional interval computation with a sufficiently high precision) if the true value of $\alpha$ is not zero.

When the true value of $\alpha$ is zero, if we carry out interval computation at an arbitrarily high precision, then the resulting interval will contain zero. Hence, as we saw in Section 1, we cannot determine zero in a finite number of steps merely with traditional intervals. However, note that the width of the resulting interval approaches 0 as the precision increases. Assuming that we can compute a quantity $\varepsilon > 0$ such that "if $\alpha$ is not zero, then $|\alpha| \geq \varepsilon$," then, we can determine whether $\alpha$ is zero or not by comparing the resulting interval with the above $\varepsilon$. This is the basis of our theory.

**Definition 4 (separation bound)** Let $\alpha$ be an algebraic number represented by a straight-line program in a certain number of given algebraic numbers. If we know a number $\varepsilon \in \mathbb{R}$, $\varepsilon > 0$, for which it is somehow assured from the representation of $\alpha$ that $|\alpha| < \varepsilon$ implies $\alpha = 0$, then we call $\varepsilon$ a separation bound for $\alpha$.

We formulate the principle in terms of the following theorem.

**Theorem 1 (principle of zero determination)** *Fix an interval arithmetic system* $\boldsymbol{I} = (K, \mathcal{I}, +, -, \times)$. *Let* $\alpha_1, \alpha_2, \ldots, \alpha_n \in K$ *be algebraic numbers and let* $\alpha$ *be an algebraic number represented in algebraic numbers* $\alpha_1, \alpha_2, \ldots, \alpha_n$ *by a straight-line program* $\boldsymbol{P}$.

*Let* $\{I_{i,\mu}\}_\mu$ *be an approximate interval sequence for* $\alpha_i$, *and let* $I_\mu$ *be the evaluation of the straight-line program* $\boldsymbol{P}$ *for the intervals* $I_{1,\mu}, I_{2,\mu}, \ldots, I_{n,\mu}$.

*Finally, we assume that we know a separation bound* $\varepsilon$ *for* $\alpha$.

1. *If there is an integer* $\mu$ *such that* $0 \notin I_\mu$ *holds then* $\alpha \neq 0$. *Furthermore, if* $\alpha$ *is real, we can determine either* $\alpha > 0$ *or* $\alpha < 0$ *from the interval* $I_\mu$. *Conversely, if* $\alpha \neq 0$, *then there is a finite precision* $\mu_0$ *such that* $0 \notin I_\mu$ *holds for any precision* $\mu \geq \mu_0$.

2. *If there is an integer* $\mu$ *such that* $0 \in I_\mu$ *and* $\max\{\, |c| \mid c \in I_\mu \,\} < \varepsilon$, *then* $\alpha = 0$. *Conversely, if* $\alpha = 0$, *then* $0 \in I_\mu$ *holds for any precision* $\mu$. *Moreover, there is a finite precision* $\mu_0$ *such that* $\max\{\, |c| \mid c \in I_\mu \,\} < \varepsilon$ *holds for any precision* $\mu \geq \mu_0$.

The proof is obvious due to the definitions of separation bounds and interval arithmetic systems.

The problem is how to compute separation bounds. Furthermore, in our setting, that is, if $\alpha$ is represented by a straight-line program, it is necessary that a separation bound for $\alpha$ can be computed from separation bounds for the initially given $\alpha_i$'s. For this, we can use various polynomial norms and heights.

For a polynomial $P(x) = \sum_{i=0}^{d} a_i x^i$, there are various norms of $P$. For example,

$$\|P\|_1 = \sum_{i=0}^{d} |a_i|, \qquad \|P\|_2 = \left(\sum_{i=0}^{d} |a_i|^2\right)^{1/2}, \qquad \|P\|_\infty = \max_{0 \le i \le d}\{|a_i|\}.$$

Let $\alpha$ be a root of $P(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{Z}[x]$, where $a_d \neq 0$. From Cauchy's inequality for the bound of roots of a polynomial,

$$|\alpha| < 1 + \frac{\max\{|a_0|, \ldots, |a_{d-1}|\}}{|a_d|}$$

(see [22] for an example), we obtain

$$|\alpha| < 1 + \|P\|_\infty.$$

If $\alpha \neq 0$, then $1/\alpha$ is a root of $Q(x) = \sum_{i=0}^{d} a_i x^{d-i}$ and $\|P\|_\infty = \|Q\|_\infty$ holds, we have

$$\left|\frac{1}{\alpha}\right| < 1 + \|P\|_\infty,$$

that is,

$$|\alpha| > \frac{1}{1 + \|P\|_\infty}.$$

We can take $\|\cdot\|_1$ or $\|\cdot\|_2$ instead of $\|\cdot\|_\infty$ since the following inequalities hold.

$$\|P\|_\infty \le \|P\|_2 \le \|P\|_1$$

However, we use the Mahler measure among various norms and heights. We describe the reason in the next section.

### 2.1.2 The Mahler Measure

Mahler defined a measure of a polynomial [19], which we will call the *Mahler measure of a polynomial.* In this section, we review the definition of the Mahler measure and its properties. We use symbols $\alpha_i$, $\beta_j$, … for conjugates for $\alpha$, $\beta$, …, instead of input algebraic numbers from here to the end of Section 2.

**Definition 5 (Mahler measure)** The Mahler measure $M(P)$ of a complex coefficient polynomial $P(x) = \sum_{i=0}^{d} a_i x^i = a_d \prod_{i=1}^{d}(x - \alpha_i)$ $(a_d \neq 0)$ is defined by the formula

$$M(P) = |a_d| \prod_{i=1}^{d} \max\{1, |\alpha_i|\}.$$

For an algebraic number $\alpha$, the Mahler measure $M(\alpha)$ of $\alpha$ is defined by the formula

$$M(\alpha) = M(P),$$

where $P$ is the primitive minimal polynomial of $\alpha$ over $\mathbb{Z}$.

From a modern point of view, the Mahler measure is a version of the height (see [33], for instance). Let $F$ be an algebraic number field, then the (logarithmic) height of an algebraic number $\alpha \in F$ is defined by

$$H_F(\alpha) = \sum_{v} \log(\max\{1, |\alpha|_v\}),$$

where the sum is over all places $v$ of $F$ and the $v$-adic valuations are normalized as usual in such a way that the product of $|\alpha|_v$ over all Archimedean $v$ is the absolute value of $N_{F/\mathbb{Q}}(\alpha)$, and the product of all $|\alpha|_v$ is equal to 1. Then, $H_F(\alpha) = \log(M(\alpha))$ for $F = \mathbb{Q}(\alpha)$.

To describe our main theorem, we divide the Mahler measure into two parts.

**Definition 6** For a polynomial $P(x) = \sum_{i=0}^{d} a_i x^i = a_d \prod_{i=1}^{d}(x - \alpha_i) \in \mathbb{C}[x]$ $(a_d \neq 0)$, the measures $M_0(P)$ and $M_1(P)$ are defined by the formulae

$$M_0(P) = |a_d|, \qquad M_1(P) = \prod_{i=1}^{d} \max\{1, |\alpha_i|\}.$$

For an algebraic number $\alpha$, the measures $M_i(\alpha)$ $(i = 0, 1)$ are defined by the formula

$$M_i(\alpha) = M_i(P),$$

where $P$ is the primitive minimal polynomial of $\alpha$ over $\mathbb{Z}$.

It is easy to see from Definition 6 that $M(P) = M_0(P)M_1(P)$ for $P \in \mathbb{C}[x]$ and $M(\alpha) = M_0(\alpha)M_1(\alpha)$ for $\alpha \in \overline{\mathbb{Q}}$.

Our main claim in this paper is that we can use Mahler measure estimates to obtain separation bounds for an algebraic number represented by a straight-line program in given algebraic numbers. This is summarized in the following theorem.

**Theorem 2** *Let $K$ be either $\mathbb{C}$ or $\mathbb{R}$, and put $K_0 = K \cap \overline{\mathbb{Q}}$.*

1. *Let $X$ be a set of predicates defined on $K_0$ for which we know*

   (a) *how to obtain a predicate $\xi \in X$ satisfied by $\alpha$ for each $\alpha \in K_0$ given in terms of a polynomial $f(x) \in \mathbb{Z}[x]$ vanishing at $\alpha$ and an interval isolating $\alpha$ from the other roots of $f(x)$,*

   (b) *how to obtain a predicate $\zeta \in X$ satisfied by $\alpha * \beta$ ($* \in \{+, -, \times\}$) from two predicates $\xi$, $\eta \in X$ satisfied by $\alpha$, $\beta \in K_0$ respectively,*

   (c) *how to obtain a separation bound for $\alpha \in K_0$ from a predicate $\xi \in X$ satisfied by $\alpha$.*

   *If $\alpha \in K_0$ be represented by a straight-line program $\boldsymbol{P}$ in $\alpha_1$, $\alpha_2$, ..., $\alpha_n \in K_0$, then by using the above computational methods, one can obtain a separation bound for $\alpha$.*

2. *Let $Y$ be all conditions of the form "$\deg \alpha \le d$ and $M(\alpha) \le A$," $d \in \mathbb{N}$ and $A \ge 1$, where $\deg \alpha = [\mathbb{Q}(\alpha) : \mathbb{Q}]$. Then $Y$ satisfies all of the three conditions for $X$ in 1.*

3. *Let $Y$ be all conditions of the form "$\deg \alpha \le d$, $M_0(\alpha) \le A_0$ and $M_1(\alpha) \le A_1$," $d \in \mathbb{N}$ and $A_0$, $A_1 \ge 1$. Then $Y$ satisfies all of the three conditions for $X$ in 1.*

4. *Fix an interval arithmetic system $(K, \mathcal{I}, +, -, \times)$. Let $Y$ be all conditions of the form "$\deg \alpha \le d$, $M_0(\alpha) \le A_0$, $M_1(\alpha) \le A_1$ and $\alpha \in I$," $d \in \mathbb{N}$, $A_0$, $A_1 \ge 1$ and $I \in \mathcal{I}$. Then $Y$ satisfies all of the three conditions for $X$ in 1.*

*Proof.* First, we prove part 1. We can obtain a predicate $\xi_i$ satisfied by $\alpha_i$ due to (a). Then, along the straight-line program $\boldsymbol{P}$, we can obtain a predicate $\xi$ satisfied by $\alpha$ from $\xi_1$, $\xi_2$, ..., $\xi_n$, due to (b). Finally, we can obtain a separation bound for $\alpha$ from $\xi$ due to (c).

Part 2 follows from Propositions 1 and 2 below.

To prove $Y$ in 3 satisfies (b), apply the following Propositions 3 and 4. The obtained separation bound is sharper than that in part 2 (see Remark 4).

Part 4 follows from part 3 and the definition of the interval arithmetic. If the precision of the interval arithmetic is sufficiently high, then the obtained separation bound is sharper than that in part 3 (see Proposition 5). ■

Hereafter, we describe propositions used in the proof for Theorem 2.

**Proposition 1** *The Mahler measure satisfies the following properties:*

1. *For any algebraic number $\alpha$, $M(\alpha) \geq 1$.*
2. *If $\alpha \neq 0$, then $1/M(\alpha) \leq |\alpha| \leq M(\alpha)$.*
3. *Let $\alpha$ and $\beta$ be two algebraic numbers whose degrees are at most $d$ and at most $e$, respectively, then,*

$$\begin{aligned} M(\alpha \pm \beta) &\leq 2^{de} M(\alpha)^e M(\beta)^d, \\ M(\alpha\beta) &\leq M(\alpha)^e M(\beta)^d. \end{aligned}$$

*Proof.* See [4] for instance. ■

We can determine whether an algebraic number $\alpha$ is zero or not by means of property 2 by taking $1/M(\alpha)$ as a separation bound for $\alpha$, and we can compute an upper bound for the Mahler measure after a ring operation among two algebraic numbers by means of property 3.

We do not necessarily have to compute approximation values of roots of $P$ to evaluate the Mahler measure of $P$.

**Proposition 2 (Landau's inequality [16])** *Among the norm $\|\cdot\|_2$ and the Mahler measure, Landau's inequality,*

$$M(P) \leq \|P\|_2, \tag{1}$$

*holds.*

**Remark 1** Due to the inequality $M(P) \leq \|P\|_2$ valid for any $P \in \mathbb{C}[x]$, one can start with $\|P\|_2$ as an upper side estimate for the Mahler measure of each input $\alpha$, where $P$ is a primitive polynomial in $\mathbb{Z}[x]$ having $\alpha$ as a root, without computing all roots of $P$.

Also, it is easy to see from the definition that $M(PQ) = M(P)M(Q)$ for any $P$, $Q \in \mathbb{C}[X]$ and that $M(P) \geq 1$ for any $P \in \mathbb{Z}[x]$. Thus one can even start with $\|P\|_2$, where $P$ is any polynomial in $\mathbb{Z}[x]$ having $\alpha$ as a root, as an estimate for $M(\alpha)$.

Suppose that we use $1 + \|\cdot\|_\infty$ instead of the Mahler measure. Let $\alpha$ and $\beta$ be algebraic numbers whose integer coefficient primitive minimal polynomials are $P_1$ and $P_2$, respectively. Then, we should estimate an upper bound for $\|Q\|_\infty$, where $Q$ is an integer coefficient polynomial that has $\alpha * \beta$ ($* \in \{+, -, \times\}$) as a root, and represent it as an expression in $\|P_1\|_\infty$ and $\|P_2\|_\infty$, which becomes more complicated.

## 2.2 Improvements of Inequalities

In this section, we improve the inequalities in the item 3 of Proposition 1 in two ways.

### 2.2.1 Improvement without using the Values of $\alpha$ and $\beta$

First we give an improvement without using the values of $\alpha$ and $\beta$.

The following proposition is clear from Definition 6.

**Proposition 3** *The measures $M_0$ and $M_1$ have the following properties:*

1. $M_i(PQ) = M_i(P)M_i(Q)$ *for* $P$, $Q \in \mathbb{C}[x]$.
2. $M_i(P) \geq 1$ *for* $P \in \mathbb{Z}[x]$.
3. $M_i(P) \geq M_i(\alpha)$ *for* $P \in \mathbb{Z}[x]$ *vanishing at* $\alpha$.

Furthermore, $M_0$ and $M_1$ have the following properties:

**Proposition 4** *Let $\alpha$ and $\beta$ be algebraic numbers of degrees at most $d$ and at most $e$, respectively. Suppose that the degree of $\alpha * \beta$, where $* \in \{+, -, \times\}$, is at most $f$. Then the following inequalities hold:*

1. $M_0(\alpha)$, $M_1(\alpha) \geq 1$.
2. $M_0(\alpha * \beta) \leq M_0(\alpha)^e M_0(\beta)^d$.
3. $M_1(\alpha\beta) \leq M_1(\alpha)^e M_1(\beta)^d, \qquad M_1(\alpha \pm \beta) \leq C.$

   *Here, $C$ is the product of the $f$ largest numbers among the following de numbers:*

$$M_1(\alpha) + M_1(\beta), \underbrace{M_1(\alpha) + 1, \ldots, M_1(\alpha) + 1}_{e-1},$$

$$\underbrace{M_1(\beta) + 1, \ldots, M_1(\beta) + 1}_{d-1}, \underbrace{2, \ldots, 2}_{(d-1)(e-1)}.$$

**Remark 2** Note that in the item 3 of Proposition 4, the following inequalities hold:

$$M_1(\alpha) + M_1(\beta) \geq M_1(\alpha) + 1 \geq 2,$$

$$M_1(\alpha) + M_1(\beta) \geq M_1(\beta) + 1 \geq 2.$$

Hence, the problem is whether $M_1(\alpha) + 1 \geq M_1(\beta) + 1$ or not.

*Proof.* The inequalities $M_i(\alpha) \geq 1$ immediately follow the definition.

Let $\alpha_1 = \alpha$, $\alpha_2$, $\dots$, $\alpha_m$ ($m \leq d$) and $\beta_1 = \beta$, $\beta_2$, $\dots$, $\beta_n$ ($n \leq e$) be all $\mathbb{Q}$-conjugates of $\alpha$ and $\beta$ respectively. Then the following polynomial

$$M_0(\alpha)^n M_0(\beta)^m \prod_{i=1}^{m}\prod_{j=1}^{n}(x - (\alpha_i * \beta_j)),$$

where $* \in \{+, -, \times\}$, is an integer coefficient polynomial that has $\alpha * \beta$ as a root. Therefore, (2) holds.

Since we have

$$\begin{aligned} M_1(\alpha\beta) &\leq \prod_{i=1}^{m}\prod_{j=1}^{n}\max\{1, |\alpha_i\beta_j|\} \\ &\leq \prod_{i=1}^{m}\prod_{j=1}^{n}(\max\{1, |\alpha_i|\} \cdot \max\{1, |\beta_j|\}) \\ &= \left(\prod_{i=1}^{m}\max\{1, |\alpha_i|\}\right)^n \left(\prod_{j=1}^{n}\max\{1, |\beta_j|\}\right)^m \\ &= M_1(\alpha)^n M_1(\beta)^m \\ &\leq M_1(\alpha)^e M_1(\beta)^d, \end{aligned}$$

the first statement in (3) holds.

For the case of addition (the case of subtraction is quite similar), we pick a subset $S$ of $\{\, (i,j) \mid 1 \leq i \leq m,\ 1 \leq j \leq n \,\}$ such that the numbers $\alpha_i + \beta_j$, $(i,j) \in S$, exhaust all $\mathbb{Q}$-conjugates of $\alpha + \beta$, with any redundancies removed. Then we have

$$\begin{aligned} M_1(\alpha + \beta) &= \prod_{(i,j)\in S}\max\{1, |\alpha_i + \beta_j|\} \\ &\leq \prod_{(i,j)\in S}(\max\{1, |\alpha_i|\} + \max\{1, |\beta_j|\}), \end{aligned}$$

and since $\#S \leq f$ holds, the second statement in (3) follows from the following lemma. ∎

**Lemma 1** *Let $S$ be a set of $s$ pairs $(i,j)$ $(1 \leq i \leq m,\ 1 \leq j \leq n)$ and put $F(x_1, \dots, x_m, y_1, \dots, y_n) = \prod_{(i,j)\in S}(x_i + y_j)$.*

*For constants $A$, $B \geq 1$, the maximum value of the continuous function $F$ on the compact set*

$$D = \left\{ (x_1, \ldots, x_m, y_1, \ldots, y_n) \in \mathbb{R}^{m+n} \;\middle|\; \begin{array}{l} \prod_{i=1}^{m} x_i = A, \quad x_i \geq 1 \;\; (i = 1, \ldots, m), \\ \prod_{j=1}^{n} y_j = B, \quad y_j \geq 1 \;\; (j = 1, \ldots, n) \end{array} \right\}$$

*is not greater than the product of the $s$ largest numbers among the following $mn$ numbers.*

$$A+B, \underbrace{A+1, \ldots, A+1}_{n-1}, \underbrace{1+B, \ldots, 1+B}_{m-1}, \underbrace{2, \ldots, 2}_{(m-1)(n-1)}.$$

(*Note that $A+B \geq A+1,\ 1+B \geq 2$.*)

**Remark 3** In some case (e.g. if $s = 1$ or $mn$), the maximum is equal to the product given in Lemma 1. In general, however, the equality does not hold. Consider the case $m$, $n > 1$, $A > B > 1$ and $S = \{(1,1), (2,2)\}$. The maximum is $(A+1)(B+1)$; on the other hand, the product given in Lemma 1 is $(A+B)(A+1)$.

*Proof.* First we prove the case where $m = 1$. Let $(a_1, b_1, \ldots, b_n)$ be a point where $F$ takes the maximum. Assume that there are two indices $p$ and $q$ satisfying the following conditions:

$$1 \leq p < q \leq n, \qquad (1,p), (1,q) \in S, \qquad b_p, b_q > 1.$$

Then we define $b'_j$'s as follows.

$$b'_j = \begin{cases} b_p b_q & \text{if } j = p; \\ 1 & \text{if } j = q; \\ b_j & \text{otherwise.} \end{cases}$$

Then $(a_1, b'_1, \ldots, b'_n) \in D$ and

$$\begin{aligned} & F(a_1, b'_1, \ldots, b'_n) - F(a_1, b_1, \ldots, b_n) \\ & \quad = \left((a_1 + b_p b_q)(a_1 + 1) - (a_1 + b_p)(a_1 + b_q)\right) \prod_{\substack{(1,j) \in S \\ j \neq p,q}} (a_1 + b_j) \\ & \quad = a_1 (b_p - 1)(b_q - 1) \prod_{\substack{(1,j) \in S \\ j \neq p,q}} (a_1 + b_j) \\ & \quad > 0, \end{aligned}$$

which contradicts the assumption that $F$ is the maximum. Hence $\#\{\, j \mid b_j > 1\,\} \leq 1$; accordingly $b_j = B$ for some $j$ and $b_{j'} = 1$ for all $j' \neq j$.

Next, we consider the general case. We write $N(k) = \#\{(i,j) \in S \mid i = k\}$. If necessary, by changing the order of the variables $x_i$, we may assume that $\{i \mid N(i) > 0\} = \{1, 2, \ldots, p\}$ and $N(1) \geq N(2) \geq \cdots \geq N(p)$.

From the above result in the case $m = 1$, we have, for each $k$ $(1 \leq k \leq p)$,

$$\prod_{\substack{(i,j)\in S \\ i=k}} (x_i + y_j) \leq (x_k + B)(x_k + 1)^{N(k)-1}$$

on $D$. Therefore,

$$\begin{aligned} F(x_1, \ldots, x_m, y_1, \ldots, y_n) &\leq \prod_{i=1}^{p} (x_i + B)(x_i + 1)^{N(i)-1} \\ &= \prod_{i=1}^{p} (x_i + B) \cdot \prod_{i=1}^{p} (x_i + 1)^{N(i)-1}. \end{aligned}$$

Again by the result for $m = 1$, we obtain

$$\prod_{i=1}^{p} (x_i + B) \leq (A + B)(1 + B)^{p-1},$$

$$\prod_{i=1}^{p} (x_i + 1)^{N(i)-1} \leq (A + 1)^{N(1)-1} \cdot 2^c, \qquad c = \sum_{i=2}^{p} (N(i) - 1).$$

That is,

$$F(x_1, \ldots, x_m, y_1, \ldots, y_n) \leq (A + B)(A + 1)^{N(1)-1}(1 + B)^{p-1} \cdot 2^c.$$

By the following equality

$$1 + (N(1) - 1) + (p - 1) + c = s,$$

and the following inequalities

$$N(1) - 1 \leq n - 1, \qquad p - 1 \leq m - 1, \qquad c \leq (m - 1)(n - 1),$$

the statement holds. ∎

**Example 2** Generically $f = de$. However, in some cases, we know $f < de$ from the expressions of $\alpha$ and $\beta$. For example, let $P_i = (x_i, y_i)$ for $i = 1, 2, 3$ be three points in $\mathbb{R}^2$. Determining whether $P_3$ is to the left of, to the right of, or on the directed line $P_1$ to $P_2$ is to decide the signature of the following determinant:

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix},$$

namely, the determinant is positive, negative or zero, depending on whether $P_3$ is to the left of, to the right of or on the directed line $P_1$ to $P_2$.

Let $d_i$ and $e_i$ be the degrees of $x_i$ and $y_i$, respectively. We write

$$\alpha = (x_2 - x_1)(y_3 - y_1), \qquad \beta = (x_3 - x_1)(y_2 - y_1).$$

Then, the degrees of $\alpha$ and $\beta$ are at most $d_1d_2e_1e_3$ and at most $d_1d_3e_1e_2$, respectively. The degree of $\alpha - \beta$ is at most $d_1d_2d_3e_1e_2e_3$; however, if we only know that $\alpha$ and $\beta$ are two algebraic numbers with degrees at most $d_1d_2e_1e_3$ and at most $d_1d_3e_1e_2$, respectively, then we can only say that the degree of $\alpha - \beta$ is at most $d_1^2d_2d_3e_1^2e_2e_3$.

For an algebraic integer $\alpha$, we have $M(\alpha) = M_1(\alpha)$. Thus we can obtain the next corollary.

**Corollary 1** *Let $\alpha$ and $\beta$ be algebraic integers of degrees at most $d$ and at most $e$, respectively. Then the following inequality holds:*

$$M(\alpha \pm \beta) \leq 2^{(d-1)(e-1)}(M(\alpha) + M(\beta))(M(\alpha) + 1)^{e-1}(M(\beta) + 1)^{d-1}$$

**Remark 4** The above inequality is a refinement of the standard inequality asserted in Proposition 1. Since $1 \leq M(\alpha), M(\beta)$, we obtain

$$\left(\frac{1}{M(\alpha)} + \frac{1}{M(\beta)}\right)\left(1 + \frac{1}{M(\alpha)}\right)^{e-1}\left(1 + \frac{1}{M(\beta)}\right)^{d-1} \leq 2^{d+e-1}.$$

By multiplying the both sides of the inequality by $2^{(d-1)(e-1)}M(\alpha)^eM(\beta)^d$, we obtain

$$2^{(d-1)(e-1)}(M(\alpha) + M(\beta))(M(\alpha) + 1)^{e-1}(M(\beta) + 1)^{d-1} \leq 2^{de}M(\alpha)^eM(\beta)^d.$$

The equality holds if and only if $M(\alpha) = M(\beta) = 1$. Note that for a nonzero algebraic integer $\gamma$, Kronecker proved that $M(\gamma) = 1$ holds if and only if $\gamma$ is a root of unity [15].

### 2.2.2 Improvement using the Values of $\alpha$ and $\beta$

If the values of some conjugates of $\alpha$ and $\beta$ are known, we can sharpen the inequalities in Propositions 4. To simplify the arguments, we only treat the case where we only know the values of $\alpha$ and $\beta$.

**Proposition 5** *Let $\alpha$ and $\beta$ be algebraic numbers of degrees at most $d$ and at most $e$, respectively. To simplify the expressions, we suppose that $d \le e$ by interchanging $\alpha$ and $\beta$ if necessary, and we thus write*

$$a = \min\{1, |\alpha|\}, \qquad b = \min\{1, |\beta|\},$$

$$A = \frac{M_1(\alpha)}{\max\{1, |\alpha|\}}, \qquad B = \frac{M_1(\beta)}{\max\{1, |\beta|\}}.$$

*Then the following inequalities hold. In 2, we assume that we know the degree of $\alpha * \beta$ (where $*$ is either $+$ or $-$) is at most $f$.*

1. $M_1(\alpha\beta) \le \max\{1, aM_1(\beta)\} \cdot \max\{1, bM_1(\alpha)\} \cdot M_1(\alpha)^{e-1} M_1(\beta)^{d-1}$.

2. $M_1(\alpha \pm \beta) \le C \cdot \max\{1, |\alpha \pm \beta|\}$, *where $C$ is as follows:*

   - *If $d = e = 1$, then $C = 1$.*
   - *If $d = 1$ and $e > 1$, then $C$ is the product of the $f - 1$ largest numbers among the following $e - 1$ numbers.*

     $$|\alpha| + B, \underbrace{|\alpha| + 1, \ldots, |\alpha| + 1}_{e-2}.$$

   - *If $d > 1$ and $e > 1$, then $C$ is the product of the $f - 1$ largest numbers among the following $de - 1$ numbers.*

     $$A + B, A + |\beta|, B + |\alpha|, \underbrace{A + 1, \ldots, A + 1}_{e-2}, \underbrace{B + 1, \ldots, B + 1}_{d-2},$$

     $$\underbrace{|\alpha| + 1, \ldots, |\alpha| + 1}_{e-2}, \underbrace{|\beta| + 1, \ldots, |\beta| + 1}_{d-2}, \underbrace{2, \ldots, 2}_{(d-2)(e-2)}.$$

*Proof.* Let $\alpha_1 = \alpha$, $\alpha_2$, …, $\alpha_m$ ($m \le d$) and $\beta_1 = \beta$, $\beta_2$, …, $\beta_n$ ($n \le e$) be all $\mathbb{Q}$-conjugates of $\alpha$ and $\beta$ respectively.

First, we will prove the inequality in 1. We distinguish several cases according to the values of $|\alpha|$ and $|\beta|$. If $|\alpha|$, $|\beta| \ge 1$, so that $a = b = 1$, this is clear. To prove the other cases, first we will show that the following inequality holds for $|\alpha| < 1$:

$$\prod_{i=1}^{n} \max\{1, |\alpha\beta_i|\} \leq \max\{1, |\alpha| M_1(\beta)\}.$$

If $|\alpha\beta_i| \leq 1$ holds for all $i$ then the statement is clear. Otherwise, there is an index $j$ such that $|\beta_j| > |\alpha\beta_j| > 1$. Therefore,

$$\begin{aligned} \prod_{i=1}^{n} \max\{1, |\alpha\beta_i|\} &= |\alpha\beta_j| \prod_{\substack{1\leq i\leq n \\ i\neq j}} \max\{1, |\alpha\beta_i|\} \\ &\leq |\alpha||\beta_j| \prod_{\substack{1\leq i\leq n \\ i\neq j}} \max\{1, |\beta_i|\} \\ &= |\alpha| \prod_{i=1}^{n} \max\{1, |\beta_i|\} \\ &= |\alpha| M_1(\beta). \end{aligned}$$

Now, we suppose that $|\alpha| < 1$ and $|\beta| \geq 1$ (the case where $|\alpha| \geq 1$ and $|\beta| < 1$ is similar). Then we have

$$\begin{aligned} M_1(\alpha\beta) &\leq \prod_{i=1}^{m}\prod_{j=1}^{n} \max\{1, |\alpha_i\beta_j|\} \\ &= \prod_{j=1}^{n} \max\{1, |\alpha\beta_j|\} \cdot \prod_{i=2}^{m}\prod_{j=1}^{n} \max\{1, |\alpha_i\beta_j|\} \\ &\leq \max\{1, aM_1(\beta)\} \cdot \prod_{i=2}^{m}\prod_{j=1}^{n} \max\{1, |\alpha_i\beta_j|\}. \end{aligned}$$

We estimate the last part:

$$\begin{aligned} \prod_{i=2}^{m}\prod_{j=1}^{n} \max\{1, |\alpha_i\beta_j|\} &\leq \prod_{i=2}^{m}\prod_{j=1}^{n} (\max\{1, |\alpha_i|\} \cdot \max\{1, |\beta_j|\}) \\ &= \prod_{i=2}^{m} \max\{1, |\alpha_i|\}^n \cdot \prod_{j=1}^{n} \max\{1, |\beta_j|\}^{m-1} \\ &= M_1(\alpha)^n M_1(\beta)^{m-1} \\ &\leq M_1(\alpha)^e M_1(\beta)^{d-1} \\ &= bM_1(\alpha) \cdot M_1(\alpha)^{e-1} M_1(\beta)^{d-1}. \end{aligned}$$

If $|\alpha| < 1$ and $|\beta| < 1$, by an argument similar to the case above, we have

$$\begin{aligned}
M_1(\alpha\beta) &\leq \prod_{i=1}^{m}\prod_{j=1}^{n}\max\{1, |\alpha_i\beta_j|\} \\
&= \prod_{j=1}^{n}\max\{1, |\alpha\beta_j|\}\cdot\prod_{i=1}^{m}\max\{1, |\alpha_i\beta|\}\cdot\prod_{i=2}^{m}\prod_{j=2}^{n}\max\{1, |\alpha_i\beta_j|\} \\
&\leq \max\{1, aM_1(\beta)\}\cdot\max\{1, bM_1(\alpha)\}\cdot M_1(\alpha)^{e-1}M_1(\beta)^{d-1}.
\end{aligned}$$

Next, we will prove 2 in the case of addition (the case of subtraction is similar). If $d = e = 1$, the statement is clear. If $d = 1$ and $e > 1$, the statement follows from Lemma 1.

For the case $d$, $e \geq 2$, first we define a set $S$ as described in the proof of Proposition 4. Then we have

$$\begin{aligned}
&M_1(\alpha+\beta) \\
&\quad= \max\{1, |\alpha+\beta|\}\cdot\prod_{\substack{(i,j)\in S\\(i,j)\neq(1,1)}}\max\{1, |\alpha_i+\beta_j|\} \\
&\quad\leq \max\{1, |\alpha+\beta|\}\cdot\prod_{\substack{(i,j)\in S\\(i,j)\neq(1,1)}}\max\{1, |\alpha_i|+|\beta_j|\} \\
&\quad\leq \max\{1, |\alpha+\beta|\}\cdot\prod_{\substack{(1,j)\in S\\j\neq1}}(|\alpha|+\max\{1, |\beta_j|\} \\
&\qquad\times\prod_{\substack{(i,1)\in S\\i\neq1}}(\max\{1, |\alpha_i|\}+|\beta|)\cdot\prod_{\substack{(i,j)\in S\\i\geq2\\j\geq2}}(\max\{1, |\alpha_i|\}+\max\{1, |\beta_j|\}).
\end{aligned}$$

We write

$$N_1 = \#\{\, j \mid (1,j)\in S\,\}, \qquad N_2 = \#\{\, i \mid (i,1)\in S\,\},$$

$$N = \#\{\,(i,j)\in S \mid i\geq 2,\, j\geq 2\,\}.$$

Then, by an argument similar to the proof of Lemma 1, we can derive

$$\begin{aligned}
&\prod_{\substack{(1,j)\in S\\j\neq1}}(|\alpha|+\max\{1, |\beta_j|\}) \\
&\qquad\leq (|\alpha|+B)(|\alpha|+1)^{N_1-2} \qquad \text{(if } N_1 = 1 \text{ this factor vanishes),}
\end{aligned}$$

$$\prod_{\substack{(i,1)\in S\\ i\neq 1}} (\max\{1,|\alpha_i|\}+|\beta|)$$
$$\leq (A+|\beta|)(1+|\beta|)^{N_2-2} \qquad \text{(if } N_2=1 \text{ this factor vanishes)},$$

$$\prod_{\substack{(i,j)\in S\\ i\geq 2\\ j\geq 2}} (\max\{1,|\alpha_i|\}+\max\{1,|\beta_j|\}) \leq C',$$

where $C'$ is the product of the $N$ largest numbers among the following $(d-1)(e-1)$ numbers.

$$A+B, \underbrace{A+1,\ldots,A+1}_{e-2}, \underbrace{1+B,\ldots,1+B}_{d-2}, \underbrace{2,\ldots,2}_{(d-2)(e-2)}.$$

The numbers of the factors of the right sides of the above inequalities are $N_1-1$, $N_2-1$ and $N$ respectively. Since

$$(N_1-1)+(N_2-1)+N = \#S-1 \leq f-1,$$

the statement holds. ■

## 3 Application to Algorithms in Computer Algebra

In this section, we discuss the practical usages of the principle in an algorithm consisting of ring operations and branchings on equality conditions (and on signature conditions if the numbers are real).

For such an algorithm, we can apply the theory of stabilizing algebraic algorithms [31]. That is, using interval computation with "zero rewriting," the rule of replacing any interval containing zero by a single point zero whenever such an interval appears in the course of computation, we can obtain the output converging to the true output as the precision increases. Moreover, at a finite precision value, a reasonable output is arrived at (for example, the degree of the output polynomial is equal to the true degree when computing the greatest common divisor of polynomials).

This approach relates to a qualitative result, namely the convergence of the final output. This means that at any step in which zero rewriting is performed, irrespective of whether the rewritten interval is truly zero, the method passes it through toward the output. Along this line, stopping at some step and asking whether a rewritten interval is truly zero has been the motivation of the research on zero determination.

Consider an algorithm consisting of ring operations and branchings on equality conditions (and on signature conditions if the numbers are real). For each input algebraic number $\alpha$, we assume that we know an integer coefficient polynomial that has $\alpha$ as a root. It is not necessary to use the minimal polynomial since for an integer coefficient polynomial $P$ vanishing at $\alpha$, the inequality $M(\alpha) \leq M(P)$ holds as described in Remark 1. It is desirable, however, to use the minimal polynomial since small upper bounds of the Mahler measures can be established.

Furthermore, we assume that an approximate value of each input algebraic number is given as an interval that has only a single root of the polynomial.

In the following, we will explain:

- How to compute the Mahler measure for each input algebraic number.
- How to store (an upper bound of) the Mahler measure.
- How to apply numeric computations with the Mahler measure computations.

To store an upper bound of the Mahler measure, we can use either rationals (or integers) or floating-point numbers. In practice, it may be useful to use floating-point numbers.

From here, we use symbols $\alpha_i$'s for input algebraic numbers instead of conjugates of $\alpha$.

## 3.1 Computing the Mahler Measures of Input Numbers

Here, we discuss methods of computing and storing the Mahler measure.

For special types of algebraic numbers, we know the exact values of the Mahler measures. For example, let $a = m/n$ be a rational number represented by an irreducible fraction. Then, we have

$$M(a) = \max\{|m|, |n|\}, \qquad M_0(a) = |n|, \qquad M_1(a) = \max\left\{1, \left|\frac{m}{n}\right|\right\}.$$

Furthermore, suppose that $\alpha$ is a square root of $a$ and it is not rational. Then, we have

$$M(\alpha) = \max\{|m|, |n|\}, \qquad M_0(\alpha) = |n|, \qquad M_1(\alpha) = \max\left\{1, \left|\frac{m}{n}\right|\right\}.$$

In general, to compute upper bounds of the Mahler measure, we may use Landau's inequality (Section 2.1.2 (1)) or its refinements if we do not want to compute the roots of a polynomial directly. For example, let $P(x) = \sum_{i=0}^{d} a_i x^i$ be a complex coefficient polynomial. Then, the following inequalities hold [21], [4]:

$$M(P) \leq \left( \|P\|_2^2 - \left| \sum_{i=0}^{d-1} a_i \bar{a}_{i+1} \right|^2 \|P\|_2^{-2} \right)^{1/2},$$

$$M(P)^2 + |a_0 a_d|^2 M(P)^{-2} \leq \|P\|^2.$$

See [4], [23] for other methods to estimate the Mahler measure.

## 3.2 Conjunction of Interval Arithmetic and Mahler Measure Estimation

When we find an interval containing zero, we need to estimate the Mahler measure corresponding to it. If the straight-line program representation is given explicitly, then there is no difficulty. However, in general, straight-line program representations are not given explicitly in real programs. Moreover, computation histories that are necessary to construct straight-line program representations are not stored. Therefore, we have proposed two methods to estimate upper bounds for the Mahler measures in real programs.

### 3.2.1 Introducing an Object Class Interval-with-Mahler-Measures

The first method is an extension of traditional interval arithmetic; we compute intervals and (upper bounds of) the Mahler measures of algebraic numbers simultaneously so that we do not have to be concerned with straight-line program representations. We have previously proposed this method in [26] and [27].

**Definition 7** *Let $K$ be either $\mathbb{C}$ or $\mathbb{R}$, fix an interval arithmetic system $(K, \mathcal{I}, +, -, \times)$.*

1. Let $\alpha \in K$ be an algebraic number of degree $d$ at most. Let $I$ be an interval containing $\alpha$ (possibly containing some conjugates of $\alpha$), and let $A$ be a real number such that $M(\alpha) \leq A$. We call the triplet $(I, d, A)$ the interval with the Mahler measure for $\alpha$, call $I$ its numeric component and call the pair of $d$ and $A$ its algebraic component.

2. The arithmetic between intervals with the Mahler measure is

$$(I, d, A) * (J, e, B) = (I * J, f, C)$$

   Here, $I * J$ ($* \in \{+, -, \times\}$) follows the interval arithmetic, $f$ is an upper bound of the degree of $\alpha * \beta$ ($f = de$ in general) and $C$ is an upper bound of the Mahler measure of $\alpha * \beta$.

Let $\alpha_1$, $\alpha_2$, $\ldots$, $\alpha_n$ be given algebraic numbers and suppose an algebraic number $\alpha$ is represented in $\alpha_i$'s by a straight-line program. Furthermore, suppose that the degree of $\alpha_i$ is at most $d_i$ and that the Mahler measure of $\alpha_i$ is at most $A_i$. We take an approximate interval sequences $\{I_{i,\mu}\}_\mu$ for each $\alpha_i$ and consider an interval with the Mahler measure $(I_{i,\mu}, d_i, A_i)$. Then, we perform the arithmetic among these intervals with the Mahler measure defined as above with the order following the straight-line program and obtain the result $(I_\mu, d, A)$; $d$ and $A$ are upper bounds of the degree and the Mahler measure of $\alpha$, and $\{I_\mu\}_\mu$ is an approximate interval sequence for $\alpha$.

If we are to use the improved inequalities described in Section 2.2, we replace $(d, A)$ by $(d, A_0, A_1)$, where $A_i$ bounds the measure $M_i$. We compute $A = A_0 A_1$ that bounds the Mahler measure only when it is needed. Furthermore, when $d = 1$, we use $(m, n)$ instead of $(A_0, A_1)$, where $m$ and $n$ are the numerator and the denominator of $\alpha$, respectively.

The usage of intervals with the Mahler measure for a real program is described as follows: Assume that we have a package of interval arithmetic with the Mahler measure, which contains a routine to compute approximate interval sequences for input algebraic numbers, and that we have a program, e.g., a program for constructing convex hulls, using exact rational arithmetic. Then, we can construct a new program with the above program as its main routine as follows:

1. We write a module that controls the precision of the interval computations.

2. We rewrite each operation, which is a ring operation or a branching on equality condition (or a branching on signature condition if the numbers are real), among rational numbers into the corresponding operation among intervals with the Mahler measure. Note that for a branching on equality condition, first we rewrite it into the branching on zero determination, then rewrite it into the corresponding operation among intervals with the Mahler measure.

3. We prepare an additional return value *UNDECIDED* other than usual *TRUE* and *FALSE* for the predicates. We rewrite each predicate of zero

determination as follows. If an algebraic number cannot be determined whether it is zero, then *UNDECIDED* is returned.

If a predicate returns *UNDECIDED* then the control module raises the precision and initiates the main routine to compute again for the same input. The new program will stop in a finite number of steps as described in Theorem 1 (see Discussion).

For computing upper bounds of the Mahler measures using floating-points, it is possible to use a given fixed precision even if the precision for the numeric component is raised, or the same precision as for the numeric component. In practice, only the first method is efficient.

We can easily make a package of intervals with the Mahler measure. Moreover, it is easy to apply the package to real programs because we can use it without changing the main structure of original programs.

### 3.2.2 Lazy Method

The method explained in Section 3.2.1 is simple, however, there are some disadvantages:

- For an algebraic number that can be determined as being nonzero simply from its numeric component, its algebraic component need not be computed.
- For an algebraic number that needs the Mahler measure to determine whether it is zero or not, the computations for the Mahler measure only need to be performed once.
- For an algebraic number that has already been determined whether it is zero or not, computations for zero determination for the number are not needed.

Therefore, a method that suspends computations until they are really needed is desirable. Such a method would resemble the so-called lazy rational arithmetic library (see [2], for instance).

To put this method in practice, we have to store the computation history. When an algebraic number cannot be determined to be zero or not, only those computations concerned with that number have to be reiterated.

To store the history, we can either use a tree or a directed acyclic graph. For their implementation, we can use the following schemes:

- Assigning a symbol to each input algebraic number and construct a straight-line program explicitly.

- Enhancing the data structure for the intervals. Each enhanced interval consists of a traditional interval and a symbolic definition. A symbolic definition is either an input algebraic number or an unevaluated expression that represents the sum, the difference, or the product of two other symbolic definitions.

We can also create a package of the lazy method and apply it to real programs without changing the main structure of the original programs. However, it should be noted that in general, the memory required to store such a computation history is extremely large. We can reduce the requirements of some programs (e.g., programs for constructing two-dimensional convex hulls) by removing any part of computation history that becomes obsolete. However, in such a case, the method is strongly dependent on each program and as a result either the structure of the original program or the package needs to be modified accordingly.

## 3.3 Examples

We implemented the methods in Section 3.2 in the Risa/Asir system (an experimental computer algebra system originally developed by Fujitsu Laboratories Limited [24]. The system is being developed at Kobe University from September 2000) on an HP9000/735 computer. Risa/Asir has big integers. We implemented the numbers and routines in the following order.

1. Big floating-point numbers with base 10 and rounding toward $+\infty$ and $-\infty$.
2. An arbitrary-precision interval arithmetic package based on floating-point arithmetic in 1.
3. Two Mahler measure computation routines that compute an upper bound of the Mahler measure for an algebraic number after performing ring operations among two algebraic numbers. One uses the original inequalities (Proposition 1), and the other uses the improved inequalities (Proposition 5).
4. The object class Interval-with-Mahler-measure.

### 3.3.1 Simple Examples

Here, we present two simple examples illustrating how the principle is used for zero determination.

**Example 3** Let $\alpha_1 = \sqrt{2}$, $\alpha_2 = \sqrt{3}$ and $\alpha_3 = \sqrt{6}$. Let $\beta = \alpha_1\alpha_2$ and $\gamma = \beta - \alpha_3$. Determine whether $\gamma$ is equal to zero.

Let $P \in \mathbb{Z}[x_1, x_2, x_3]$ be $x_1x_2 - x_3$. Then $\gamma = P(\alpha_1, \alpha_2, \alpha_3)$ is represented by a straight-line program $(P_1, P_2, P_3, P_4, P_5)$, where

$$P_1 = x_1, \qquad P_2 = x_2, \qquad P_3 = x_3, \qquad P_4 = P_1 \times P_2, \qquad P_5 = P_4 - P_3.$$

We use the triplet structure for intervals with the Mahler measure, using the original inequalities. For the numeric component, we use floating-point arithmetic with base 10 and we take approximate interval sequences for $\alpha_i$'s as described in Example 1.

First, we set the precision to 10. The triplet corresponding to $\alpha_1$, $\alpha_2$ and $\alpha_3$ are as follows:

$$\begin{gathered}([1.414213562, 1.414213563], 2, 2),\\ ([1.732050807, 1.732050808], 2, 3),\\ ([2.449489742, 2.449489743], 2, 6).\end{gathered}$$

The triplet for $\beta = \alpha_1 \cdot \alpha_2$ is

$$([2.449489741, 2.449489745], 4, 36),$$

and the triplet for $\gamma = \beta - \alpha_3$ is

$$([-0.2 \times 10^{-8}, 0.3 \times 10^{-8}], 8, 429981696).$$

Since the numeric component $[-0.2\times10^{-8},\ 0.3\times10^{-8}]$ of the resulting triplet contains 0, we apply Theorem 1. At a precision of 10, however, we cannot determine $\gamma$ to be zero because

$$0.3 \times 10^{-8} > \frac{1}{429981696} = 0.2325\ldots \times 10^{-8}.$$

Next, we set the precision to 11. The triplet corresponding to $\alpha_1$, $\alpha_2$ and $\alpha_3$ are as follows:

$$\begin{gathered}([1.4142135623, 1.4142135624], 2, 2),\\ ([1.7320508075, 1.7320508076], 2, 3),\\ ([2.4494897427, 2.4494897428], 2, 6).\end{gathered}$$

The triplet for $\beta = \alpha_1 \cdot \alpha_2$ is

$$([2.4494897425, 2.4494897429], 4, 36),$$

and the triplet for $\gamma = \beta - \alpha_3$ is

$$([-0.3 \times 10^{-9}, 0.2 \times 10^{-9}], 8, 429981696).$$

Since the numeric component $[-0.3 \times 10^{-9}, 0.2 \times 10^{-9}]$ of the resulting triplet contains 0, we apply Theorem 1. At a precision of 11, we can determine $\gamma$ to be truly equal to zero because

$$0.3 \times 10^{-9} < \frac{1}{429981696} = 0.2325\ldots \times 10^{-8}.$$

In this example, to estimate the Mahler measure $M(\gamma)$ is to estimate the Mahler measure of a polynomial $x^4(x^2-24)^2$, which is 576. We will show how the improvements in the inequalities result in better estimates for $M(\gamma)$. As described above, the original inequalities show that $M(\gamma) \leq 429981696$.

If we use the improved inequalities in Proposition 4, we obtain

$$M(\gamma) \leq 2^3(36+6)(36+1)(1+6)^3 = 4264176.$$

If we use the values of the algebraic numbers (Proposition 5), we obtain $M(\gamma) < 203925.8989$. Here, to compute upper bounds of the Mahler measures, we use floating-point computations with base 10, a precision of 10 and rounding toward $+\infty$, as described below:

$$A = \frac{M(\alpha_3)}{\max\{1, |\alpha_3|\}} < \frac{6}{2.449489742} < 2.449489744,$$
$$B = \frac{M(\beta)}{\max\{1, |\beta|\}} < \frac{36}{2.449489741} < 14.69693847,$$
$$\max\{1, |\gamma|\} = 1,$$

$$\begin{aligned}
M(\gamma) &\leq \max\{1, |\gamma|\}(B+A)(B+|\alpha_3|)(|\beta|+A)(1+A)^2(1+|\alpha_3|)^2 \\
&< 1 \cdot (14.69693847 + 2.449489744) \\
&\quad \times (14.69693847 + 2.449489743) \cdot (2.449489745 + 2.449489744) \\
&\quad \times (1 + 2.449489744)^2 \cdot (1 + 2.449489743)^2 \\
&< 203925.8989.
\end{aligned}$$

**Example 4** Consider the following three polynomials:

$$\begin{aligned}
f(x) &= x^5 - x + 1, \\
g(x) &= x^5 - 10x^4 + 40x^3 - 80x^2 + 79x - 29, \\
h(x) &= x^5 - 5x^4 + 8x^3 - 10x^2 + 36x - 1.
\end{aligned}$$

Each of the equations $f(x) = 0$, $g(x) = 0$ and $h(x) = 0$ has only one real root. We denote them $\alpha$, $\beta$ and $\gamma$, respectively. Determine whether $\alpha\beta - \gamma$ is equal to $-1$.

Approximate values of $\alpha$, $\beta$ and $\gamma$ are as follows:

$$\alpha = -1.1673039\ldots, \qquad \beta = 0.8326960\ldots, \qquad \gamma = 0.0279906\ldots.$$

Therefore, $\alpha\beta = -0.97200\ldots$ and $\alpha\beta - \gamma$ is almost equal to $-1$. Indeed, interval computation for $\delta = \alpha\beta - \gamma + 1$ in decimals with precision 200 is $[-1 \times 10^{-199}, 8 \times 10^{-200}]$. We will confirm that $\delta$ is exactly equal to 0 using the principle.

At first, we make a few remarks. The polynomial $f$ is irreducible in $\mathbb{Z}[x]$ and its roots cannot be obtained by applying arithmetic operations and radicals to the coefficients since the Galois group of $f$ is $S_5$, the symmetric group of degree 5.

*Outline of the proof.* The polynomial $f$ is irreducible over $\mathbb{Z}/3\mathbb{Z}$ and is decomposed over $\mathbb{Z}/2\mathbb{Z}$ in two irreducible factors whose degrees are two and three, respectively.

$$f(x) \equiv (x^2 + x + 1)(x^3 + x^2 + 1) \pmod 2$$

Therefore, $f$ is irreducible in $\mathbb{Z}[x]$, and the Galois group $G$ of $f$ contains a cycle of length 5 and a permutation of the type $(i\,j)(k\,l\,m)$. These facts imply that $G$ is isomorphic to $S_5$ (see [32], for example). ■

First, we estimate the Mahler measure of $\alpha$, $\beta$ and $\gamma$. Using Landau's inequality, we estimate the Mahler measures as follows:

$$\begin{aligned} M(\alpha) &\leq \sqrt{3} < 2, \\ M(\beta) &\leq \sqrt{15183} < 124, \\ M(\gamma) &\leq \sqrt{1487} < 39. \end{aligned}$$

For simplicity, we use integer values as the estimation.

- Estimation by the original method:

$$M(\delta) < 7.859\ldots \times 10^{174}$$

- Estimation by the first improvement (Proposition 4):

$$M(\delta) < 3.446\ldots \times 10^{164}$$

- Estimation by the second improvement (Proposition 5):

$$M(\delta) < 7.344\ldots \times 10^{112}$$

In any case, we can decide that $\delta = 0$ by the above estimation of the Mahler measure for $\delta$ and by the following inequalities:

$$-1 \times 10^{-199} < \delta < 8 \times 10^{-200}.$$

**Remark 5** The polynomials $f$, $g$ and $h$ have the following relations:

$$g(x) = f(x-2), \qquad h(x^2) = -f(x-1)f(-x-1).$$

That is, $\beta = \alpha + 2$, $\gamma = (\alpha+1)^2$, therefore,

$$\delta = \alpha\beta - \gamma + 1 = \alpha(\alpha+2) - (\alpha+1)^2 + 1 = 0.$$

### 3.3.2 Application to Graham's Algorithm

We applied the principle to construct two-dimensional convex hulls. The convex hull of a set $S$ of points is the smallest convex set containing $S$. Hereafter, we assume that $S$ is a finite set in $\mathbb{R}^2$ so that the convex hull of $S$ is a convex polygon. Therefore, for a given finite set $S$ of points, "to find the convex hull of $S$" means "to give an ordered list of vertices of the convex hull of $S$."

There are several well-known algorithms for constructing two-dimensional convex hulls such as Graham's algorithm, Jarvis' algorithm and Bentley-Shamos' algorithm (see [25] for instance). A basic routine in each algorithm is to determine whether $P_3$ is to the left of, to the right of, or on the directed line from $P_1$ to $P_2$, for three given points $P_1$, $P_2$ and $P_3$. That is, to determine the signature of the determinant described in Example 2. Graham's algorithm is as follows.

**Graham's algorithm.**
**Input:** A finite set $S = \{P_1, P_2, \ldots, P_m\} \subset \mathbb{R}^2$.
**Output:** A list of vertices for the convex hull of $S$.

1. [Base point] Pick a point in $S$ that will be a vertex of the convex hull (e.g., the point with the largest $y$-coordinate among the points with the smallest $x$-coordinate) and call it $O$.

2. [Sort] Sort the points in $S$, other than $O$, in the increasing order of the arguments of the line segments from $O$ to $P_i$. If more than one points have the same argument, select the one farthest from $O$ and discard the others among them. Let $Q_1$, $Q_2$, $\ldots$, $Q_n$ be the result of this operation, and put $Q_0 = O$.

3. [Sweep] Create the convex hull by determining whether $Q_i$ is to the left of the directed line $Q_{i-2}$ to $Q_{i-1}$ or not. More precisely, the procedure can be described as follows:

```
$j := 0$
**for** $i$ **from** $0$ **to** $n$ **do**
    $j := j + 1$
    $R_j := Q_i$
    **while** $j \geq 4$ **and** $R_j$ is to the right of or on $\overrightarrow{R_{j-2}R_{j-1}}$ **do**
        $R_{j-1} := R_j$
        $j := j - 1$
**return** $[R_1, R_2, \ldots, R_j]$
```

We applied Graham's algorithm to the next example through the implementation of our methods.

**Example 5** $S$ is a set of 1000 points. Each point has a coordinate of the form $(\sqrt{X}, \sqrt{Y})$, where $X$ and $Y$ are randomly generated integers satisfying the following conditions:

$$0 \leq X \leq 100, \qquad 0 \leq Y \leq 100, \qquad X + Y \leq 100, \qquad Y \leq 3X.$$

That is, the points are bounded by a sector whose center is at the origin, has radius of 10, and the angle subtended by the arc at the center is $\pi/3$. The input points and the resulting convex hull (a polygon with 21 vertices) are described in Figure 1. We chose these conditions so that a lot of numbers would be decided as zero and the convex hull would have many vertices.

To compute a determinant

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1), \tag{2}$$

we used the order of operations as described on the right-hand side, or more precisely, the straight-line program representation $(D_1, D_2, \ldots, D_{13})$ as follows:

$$D_i = x_i, \qquad D_{i+3} = y_i \quad (i = 1,\ 2,\ 3),$$

$$D_7 = D_2 - D_1, \qquad D_8 = D_5 - D_4, \qquad D_9 = D_3 - D_1, \qquad D_{10} = D_6 - D_4,$$

$$D_{11} = D_7 \times D_{10}, \qquad D_{12} = D_8 \times D_9, \qquad D_{13} = D_{11} - D_{12}.$$

We conducted experiments on the following three methods:

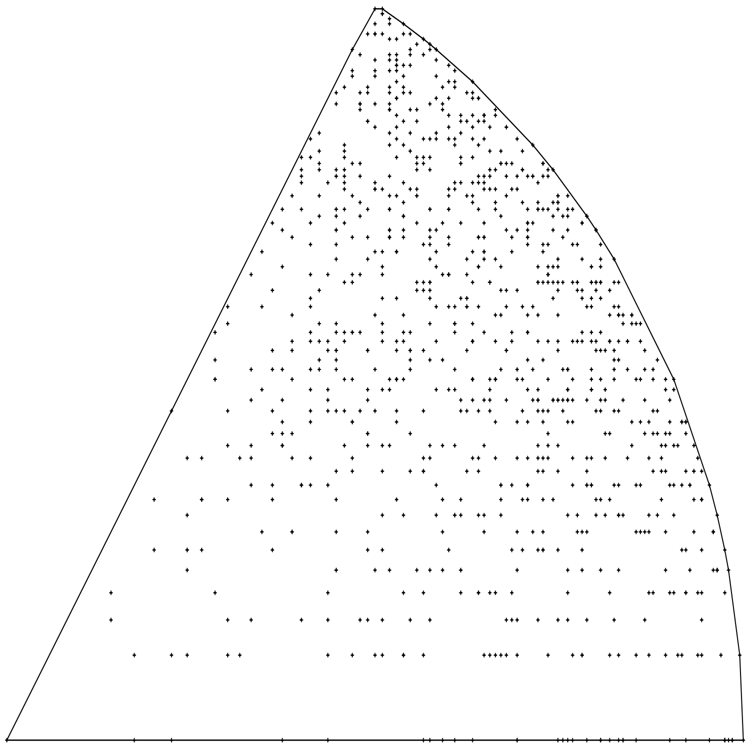

Figure 1: Input points and the convex hull for Example 5.

**Method 1:** Interval arithmetic along with Mahler measure estimation using the original inequalities in Proposition 1.

**Method 2:** Interval arithmetic along with Mahler measure estimation using the improved inequalities in Proposition 5.

**Method 3:** Lazy method.

In all of the three methods, for computing the values of algebraic numbers, initially we set the precision to 10. In Methods 1 and 2, we iteratively doubled the precision until the convex hull was obtained. In Method 3, we iteratively doubled the precision until the signature of each number was determined and

Table 1: Computation times and maximal precisions.

| | Method 1 | Method 2 | Method 3 |
|---|---|---|---|
| maximal precision | 2560 | 1280 | 1280 |
| CPU time (sec.) | $3.17 \times 10^4$ | $2.71 \times 10^4$ | 304 |
| GC time (sec.) | $6.37 \times 10^3$ | $6.90 \times 10^3$ | 157 |

after the signature was determined, we reset the precision to 10. To estimate the Mahler measure, we used big integers in Method 1 and floating-point numbers with precision of 10 digits in Methods 2 and 3.

In Methods 1 and 2, we did not take into consideration the decrease in the degrees of algebraic numbers occurring in the computation of the determinant 2. However, in Method 3, we took it into consideration.

In Method 3, we assigned indices to input algebraic numbers and changed the arguments from numbers to these indices in each subroutine containing a signature determination so that we could have straight-line program representations.

The maximal precisions in the computations, CPU times, and garbage collection (GC) times are described in Table 1. Comparing Method 1 with Method 2, the maximal precision and computation time decreased when using Method 2 due to the improvements achieved in inequalities. By omitting obsolete computations, the computation time was decreased drastically when using Method 3. The lazy method is efficient if that part of the computation history which has become obsolete can be removed efficiently. In this example, by storing only the required computation history, as described above, we achieved efficiency. If we used a package of the lazy method, the memory requirements would become extremely large.

It was confirmed that there were cases where the principle actually contributed to zero determination: in the experiment with Method 3, there were 389 such cases; this number is the same as the number of cases where the maximal precision was required in Methods 1 and 2 (2560 in Method 1 and 1280 in Method 2).

Details of precision used in Method 3 for zero determination are described in Table 2.

Table 2: The number of cases in which algebraic numbers were determined as zero (Method 3).

| precision | Base point | Sort | Sweep | total |
|---|---|---|---|---|
| 10 | 1 | 129 | 8 | 138 |
| 20 | 0 | 43 | 0 | 43 |
| 40 | 0 | 171 | 0 | 171 |
| 80 | 0 | 14 | 0 | 14 |
| 160 | 0 | 0 | 0 | 0 |
| 320 | 0 | 0 | 0 | 0 |
| 640 | 0 | 0 | 10 | 10 |
| 1280 | 0 | 0 | 13 | 13 |
| total | 1 | 357 | 31 | 389 |
| CPU time (sec.) | 0.2 | 22.4 | 277.1 | 299.7* |
| GC time (sec.) | 0.1 | 12.1 | 142.1 | 154.7* |

$*$: omitting preprocess time.

## 4 Discussion

### 4.1 Exact Computation for Algebraic Numbers

We have proposed a computation method with algebraic numbers, using approximation.

There are several ways to perform arithmetic operations and zero determination on algebraic numbers exactly. It is possible to divide them into two distinct groups. The first group represents $\alpha$ as $f(\theta)$, where $f$ is a rational coefficient polynomial and $\theta$ is a fixed algebraic number. The second group represents $\alpha$ as a root of an integer coefficient square-free polynomial. In the second group, to distinguish $\alpha$ from the other roots of the polynomial, additional information is needed. There are at least two kinds of such information. Therefore, there are at least three methods of representing an algebraic number $\alpha$.

1. In terms of a rational coefficient polynomial $f$ and a fixed algebraic number $\theta$ and representing $\alpha$ as $f(\theta)$ [5], [18].

2. Using an integer coefficient square-free polynomial $P$ having $\alpha$ as a root and an interval containing $\alpha$ but not containing the other roots of $P$ [5].

3. (For a real algebraic number) Thom's code: using an integer coefficient square-free polynomial $P$ having $\alpha$ as a root and the signatures of the derivatives $P^{(i)}$ of $P$ at $\alpha$, for $i = 1, \ldots, \deg(P) - 1$ as proposed in [9].

The associated methods to determine whether $\alpha$ is zero or not can be considered as follows:

1. Refining (if necessary) the interval containing $\theta$ [18].

2. Refining (if necessary) the interval containing $\alpha$.

3. Using a generalized Sturm algorithm for several inequalities [9].

For polynomial root isolation, see [6] (for real root isolation, see [8], [7]). If we use either the second or the third representation method, then we have to compute a square-free polynomial that has the resulting algebraic number as a root after every arithmetic operation of two algebraic numbers. The computational cost of such a polynomial is very expensive if algebraic numbers have high degrees. Therefore, the first method is preferable since we can perform arithmetic operations simply. In general, however, it is also very expensive to find such a primitive element as $\theta$ in advance. See [5], [18], for example, on exact computation among algebraic numbers.

## 4.2 Future Directions

In the course of computation using the principle for zero determination, intervals always contain the true values even after some intervals rewritten into zero. Therefore, it can be proved that if an algorithm consisting of ring operations and branchings on equality conditions (and on signature conditions if the numbers are real), with exact computation, stops in a finite number of steps for an input, then so does the rewritten algorithm. To prove the statement, use induction on the number of zero determination (and signature condition) in the course of exact computation.

Theoretically, one future direction of research is to estimate the precision needed to determine zero correctly before computation. Hiyoshi's work [11], [12] is the first attempt in this direction. Practically, future directions are for the implementation of a package of the lazy method and to carry out experiments using it.

## Acknowledgment

The author would like to thank Professor Nobuki Takayama of Kobe University for giving a motive for the research. The author is also thankful to Professor Kazuya Kato of Kyoto University, Professor Takayuki Oda of the University of Tokyo, Professor Junjiro Noguchi of the University of Tokyo, Professor Ryu Hasegawa of the University Tokyo and Professor Masayuki Noro of Kobe University for their encouragement.

Throughout this paper, the author wishes to express his thanks to the University of Tokyo for the management of this thesis. Special thanks to Professor Toshio Oshima for his many efforts to organize the procedure of the doctoral examination and Professor Itaru Terada for his valuable comments and kindly support. Finally, the author would like to appreciate the kindly support and encouragement for writing this thesis by Nippon Telegraph and Telephone Corporation, especially Dr. Kiyoshi Shirayanagi, the leader of our research group.

## References

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations, Computer Science and Applied Mathematics*, Academic Press, 1983.

[2] M. Benouamer, D. Michelucci and B. Peroche, *Error-free boundary evaluation using lazy rational arithmetic: a detailed implementation*, Proc. 2nd Symposium on Solid Modeling and Applications, pp. 115–126, 1993.

[3] C. Burnikel, R. Fleischer, K. Mehlhorn and S. Schirra, *A strong and easily computable separation bound for arithmetic expressions involving radicals*, Algorithmica, **27**, pp. 87–99, 2000.

[4] L. Cerlienco, M. Mignotte and F. Piras, *Computing the measure of a polynomial*, J. Symbolic Computation, **4**, pp. 21–33, 1987.

[5] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer-Verlag, 1993.

[6] G. E. Collins and W. Krandick, *An efficient algorithm for infallible polynomial complex root isolation*, Proc. 1992 International Symposium on Symbolic and Algebraic Computation (ISSAC'92), pp. 189–194, 1992.

[7] G. E. Collins and W. Krandick, *A hybrid method for high precision calculation of polynomial real roots*, Proc. 1993 International Symposium on Symbolic and Algebraic Computation (ISSAC'93), pp. 47–52, 1993.

[8] G. E. Collins and R. Loos, *Real zeros of polynomials*, Computer Algebra Symbolic and Algebraic Computation, B. Buchberger, G. E. Collins and R. Loos (eds.), Springer-Verlag, 1983, pp. 83–94.

[9] M. Coste and M.-F. Roy, *Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets*, J. Symbolic Computation, **5**, pp. 121–129, 1988.

[10] J. Heintz, *On the computational complexity of polynomials and bilinear mappings. A survey*, Proc. 5th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-5), Springer LNCS 356, pp. 269–300, 1989.

[11] H. Hiyoshi, *Applications of approximate computation to computational algebra and computational geometry* (*in Japanese*), Master thesis, Univ. Tokyo, 1997.

[12] H. Hiyoshi and H. Sekigawa, *Deciding the sign using floating-point arithmetic*, Abstracts of the 14th European Workshop on Computational Geometry (CG'98), pp. 89–91, 1998.

[13] J. R. Johnson, *Real algebraic number computation using interval arithmetic*, Proc. 1992 International Symposium on Symbolic and Algebraic Computation (ISSAC'92), pp. 195–205, 1992.

[14] T. Krick and L. M. Pardo, *A computational method for Diophantine approximation*, Algorithms in Algebraic Geometry and Applications, L. González-Vega, T. Recio (eds.), Birkhäuser, 1996, pp. 193–253.

[15] L. Kronecker, *Zwei Sätze über Gleichungen mit ganzzahligen Coefficienten*, J. für und angew. Math., **53**, pp. 173–175, 1857.

[16] E. Landau, *Sur quelques théorèmes de M. Petrovic relatifs aux zéros des fonctions analytiques*, Bull. Soc. Math. France, **33**, pp. 251–261, 1905.

[17] C. Li and C. Yap, *A new constructive root bound for algebraic expressions*, Proc. the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), pp. 496–505, 2001.

[18] R. Loos, *Computing in algebraic extensions*, Computer Algebra Symbolic and Algebraic Computation, B. Buchberger, G. E. Collins, R. Loos (eds.), Springer-Verlag, 1983, pp. 173–187.

[19] K. Mahler, *An application of Jensen's formulae to polynomials*, Mathematica, **7**, pp. 98–100, 1960.

[20] K. Mehlhorn and S. Schirra, *A generalized and improved constructive separation bound for real algebraic expressions*, Technical Report MPI-I-2000-004, 2000, Max-Planck-Institut für Informatik.

[21] M. Mignotte, *An inequality about factors of polynomials*, Math. Comp., **28**, pp. 1153–1157, 1974.

[22] M. Mignotte, *Some useful bounds*, Computer Algebra Symbolic and Algebraic Computation, B. Buchberger, G. E. Collins, R. Loos (eds.), Springer-Verlag, 1983, pp. 259–263.

[23] M. Mignotte and P. Glesser, *Landau's inequality via Hadamard's*, J. Symbolic Computation, **18**, pp. 179–383, 1994.

[24] M. Noro and T. Takeshima, *Risa/Asir—A computer algebra system*, Proc. 1992 International Symposium on Symbolic and Algebraic Computation (ISSAC'92), pp. 387–396, 1992.

[25] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[26] H. Sekigawa, *An interval arithmetic with algebraic complexity to determine the signs of algebraic expressions*, Abstracts of the 4th International Symposium on Effective Methods in Algebraic Geometry (MEGA'96), p. 43, 1996.

[27] H. Sekigawa, *Using interval arithmetic and polynomial norms to determine signs of algebraic numbers*, Proc. 1996 Asian Symposium on Computer Mathematics, pp. 43–53, 1996.

[28] H. Sekigawa, *Using interval computation with the Mahler measure for zero determination of algebraic numbers*, Josai Information Sciences Researches, **9**(1), pp. 83–99, 1998.

[29] K. Shirayanagi, *An algorithms to compute floating point Gröber bases*, Mathematical Computation with Maple V: Ideas and Applications, T. Lee (ed.), Birkhäuser, 1993, pp. 95–106.

[30] K. Shirayanagi, *Floating point Gröbner bases*, Mathematics and Computers in Simulation, **42**, pp. 509–528, 1996.

[31] K. Shirayanagi and M. Sweedler, *A theory of stabilizing algebraic algorithms*, Technical Report 95-28, 1995, Mathematical Sciences Institute, Cornell University.

[32] B. L. van der Waerden, *Moderne Algebra* (*8th ed.*), Springer-Verlag, 1971.

[33] D. Zagier, *Algebraic numbers close to both* 0 *and* 1, Math. Comp., **61**, pp. 485–491, 1993.