Constructive Root Bound for k-Ary Rational Input Numbers

Sylvain Pion [1]

INRIA Sophia-Antipolis,
BP 93, 06902 Sophia-Antipolis cedex, France.

Chee K. Yap

Courant Institute of Mathematical Sciences, New York University
New York, NY 10012, USA

Abstract

Constructive root bounds is the fundamental technique needed to achieve guaranteed accuracy. Guaranteeing accuracy is the critical capability in Exact Geometric Computation, an important paradigm for constructing robust geometric algorithms. Current bounds are overly pessimistic in the presense of general rational input numbers. In this paper, we introduce a method which greatly improves the known bounds for k-ary rational input numbers. Since majority of input numbers in scientific and engineering applications are such numbers, this could lead to a significant speedup for a large class of applications. We apply our method to the BFMSS Bound and the Degree-Measure Bound. Implementation and experimental results based on the `Core Library` are reported.

Key words: Constructive root bounds, exact geometric computation, robust numerical algorithms, k-ary rational numbers

1 Introduction

The critical idea of the Exact Geometric Computation (EGC) approach to robust geometric algorithms is "geometric exactness". This amounts to ensur-

Email addresses: `Sylvain.Pion@sophia.inria.fr` (Sylvain Pion), `yap@cs.nyu.edu` (Chee K. Yap).

[1] This research is supported by NSF/ITR Grant #CCR-0082056. Sylvain's work is conducted under a postdoc fellowship of this grant.

ing that all computational decisions in a program are error free. It translates to the ability to guarantee the sign of real numerical quantities. Guaranteeing the sign is a special form of "guaranteed accuracy computation" [16]. In guaranteed accuracy computation, we can pre-specify an accuracy for each numerical quantity. Guaranteeing the sign of a number amounts to guaranteeing one relative bit of the number. Such techniques have been encoded into two general libraries `LEDA_real` [6,1] and `Core Library` [4,5]. To ensure this form of numerical control, the use of root bounds is central. But classical root bounds (e.g., [9]) may be highly non-constructive. What we need are called **constructive root bounds** in [8]. Such bounds are defined relative to some set $\mathcal{E}$ of algebraic expressions. It is constructive in two ways: (i) First, for each expression $E \in \mathcal{E}$, we define a set of mutually recursive parameters $u_1(E), \ldots, u_m(E)$ (ii) Second, there is an explicit computable **root bound function** $\beta(u_1, \ldots, u_m)$ such that if E is well-defined and $E \neq 0$, then

$$|E| \geq \beta(u_1(E), \ldots, u_m(E)). \tag{1}$$

We will write $\beta(E)$ instead of $\beta(u_1(E), \ldots, u_m(E))$. To be more precise, we may call β an **exclusion** root bound; if the inequality in (1) were reversed, we would have an **inclusion** root bound.

The first examples of such constructive root bounds is Mignotte's constructive Measure Bound [10], applied to the problem of "identifying algebraic numbers". The measure bound has been sharpened by Sekigawa [14]. In EGC, such bounds were first introduced in the `Real/Expr` Package [17], where the degree-height bounds [17] and degree-length bounds [15, p. 177] were used. Scheinerman [12] gave a constructive bound for algebraic integers based on eigenvalues. Burnikel et al [2] introduced the BFMS Bound that turns out to be extremely effective for division-free expressions. Recently, this bound was improved to what we will call [2] the BFMSS Bound [3]. In [8,7], we introduced another constructive root bound that overcomes some of the shortcomings of BFMS. If β, β' are root bound functions, we can compare them in two ways: (i) efficiency and (ii) effectiveness. Efficiency refers to the complexity of computing the root bounds, and effectivity refers to the size of the bounds (a larger $\beta(E)$ is more effective). Generally, the most interesting comparison is based on effectiveness (efficiency is less of an issue in most applications because the running time is usually dominated by the multiprecision arithmetic). If $\beta'(E) \geq \beta(E)$ for all $E \in \mathcal{E}$, we say β' **dominates** β (over $\mathcal{E}$). Among the current constructive root bounds, there are three that are not dominated by any others over the class of constructible expressions: degree-measure [10,2], BFMSS [3] and Li-Yap [8]. We give a comparison of the effectiveness of these three root bounds in Section 6.

[2] The BFMS and BFMSS bounds are both named after the initials of their authors.

The starting point of this paper is the observation that (a) current constructive bounds are quite effective for division-free input expressions involving only integer inputs, and (b) the bounds become considerably worse in the presense of division. Even when the expression is division-free, the presence of rational input numbers counts as introducing division into the expression. Such ineffective bounds can make some computations impractical. We note that these ineffective bounds are sometimes intrinsic, because it is easy to see that the worst case requires exponential bit sizes. Fortunately, this is not the end of the story. The vast majority of numerical input in scientific and engineering applications involves k-ary rationals for some integer $k \geq 2$. Invariably $k = 2$ (binary) or $k = 10$ (decimal). By a **k-ary rational** we mean a rational number whose denominator is a power of k. Thus k-ary rationals are generalizations of integers.

We shall introduce a general technique that can take advantage of k-ary rationals. The technique seems orthogonal to previous techniques in the sense that for any current constructive root bound β, we can modify it to a "k-ary version" β_k which is more effective. In this paper, we introduce the k-ary version of the BFMSS and Measure Bounds. These will be referred to as the BFMSS[k] and Measure[k] Bounds. In algorithms, especially in computer algebra, it is a well-known phenomenon that rational number arithmetic is much slower than integer arithmetic. Furthermore, k-ary rational number arithmetic has a complexity that is intermediate between these two extremes. The techniques of this paper will yield the same kind of intermediate complexity for root bounds of expressions with k-ary input numbers.

Some Examples. We briefly illustrate the possible improvements with our new technique. Instead of the root bound $\beta(E)$, we normally consider the corresponding **bit-bound**, defined as $-\lg \beta(E)$.

An example from [8] is the identically zero expression $E_1(x, y) = \sqrt{x} + \sqrt{y} - \sqrt{x + y + 2\sqrt{xy}}$. Suppose x, y are L-bit binary numbers (i.e., numerator are L-bit integers and denominator are L-bit powers of 2). Table 1 compares some bit-bounds and timings (Cf. [3]). Line 1 gives the bit-bound as a function of L. Line 2 gives the range of bit-bounds computed by our Core Library implementation when 10 random choices of machine doubles are substituted for x and y. Line 3 gives the time to evaluate the 10 random examples of Line 2 for 100 times each.

When x, y are rational numbers whose numerator and denominator are L-bit integers, the Bit-Bound functions for BFMSS and Li-Yap are just $96L+30$ and $28L + 60$ (as in Line 1) while BFMSS[2] drops to $8L + 30$. On the other hand, when x, y are L-bit integers, the Bit-Bound function for all three methods is the same and equal to $7.5L+30$. This example illustrates our previous remark, that our new bit-bounds for k-ary input numbers lie between the bit-bounds

Table 1
Comparison of BFMSS, Li-Yap and BFMSS[2]

	Method	BFMSS	Li-Yap	BFMSS[2]
1	Bit-Bound function	$96L + 30$	$28L + 60$	$8L + 30$
2	Bit-Bound Range ($L = 53$)	4926-5118	2085-2165	426-462
3	Timing ($L = 53$, 100×10 times)	46.7 s	8.35 s	3.58 s

for integers and for rational numbers. Indeed, they are only slightly worse than the integer case.

Next, consider the important and common situation of evaluating $n \times n$ determinants where the input numbers are L-bit binary numbers. Such numbers have the form $m2^{-k}$ where $|m| < 2^L$ and $0 \leq k \leq L$. Let E_0 be an expression for such a determinant. First, assume E_0 is the co-factor expansion of the determinant (this is a polynomial with $n!$ terms). Then the BFMSS Bound for E_0 gives a root-bit bound that is more than

$$(n!)nL. \tag{2}$$

This is exponentially worse in n than our binary version of the BFMSS Bound, which gives a root-bit bound of $2nL$.

In our experiments (Section 6), we use a more efficient determinant expression: let E_1 be the determinant expression obtained by using dynamic programming principles. Thus E_1 is a DAG while E_0 is a tree. E.g., when the input is a random 5×5 matrix and $L = 100$, our BFMSS implementation gives the bound $-\lg |E| \geq 10,282$, while our binary version of BFMSS gives $-\lg |E| \geq 326$.

Overview. Section 2 gives a high-level view of what our k-ary transformation does to any constructive root bound. Section 3 reviews the BFMSS Bound, while Section 4 gives the new BFMSS[k] Bound. We show that BFMSS[k] dominates BFMSS. Section 5 gives the new Measure[k] Bound, and again we show that Measure[k] dominates Measure. Experiments and comparisons are given in Section 6. We conclude in Section 7.

2 Generic k-Ary Method

We propose a meta-method for exploiting k-ary input numbers. The meta-method is applicable to any constructive root bounding method. In particular, we will apply it to the BFMSS Bound and the Measure Bound. In general, if β is a root bound function as in (1), our k-ary transformation produces a related root bound function β_k. If $\beta^{\rm bfmss}$ and $\beta^{\rm meas}$ are the root bound functions corresponding to the BFMSS and Measure Bounds, we will describe their binary version are $\beta_2^{\rm bfmss}$ and $\beta_2^{\rm meas}$.

As usual, we consider the class of expressions which are DAGs with rational numbers at the leaves and whose internal nodes are algebraic operators. The typical class of algebraic operators are $+, -, \times, \div$ and algebraic root extraction, but this may vary depending on context. Let $\mathrm{val}(E)$ be the algebraic number denoted by E. Since algebraic operators are partial functions, $\mathrm{val}(E)$ may be undefined. In any inequality involving $\mathrm{val}(E)$, it is understood that the inequality is in effect only when both sides are defined. We usually write "E" instead of $\mathrm{val}(E)$ when this is clear from context.

The basic idea of the k-ary transformation is to transform an expression E to another expression E_k such that

$$E = k^{v(E)} E_k \tag{3}$$

and $v(E) = v_k(E) \in \mathbb{Z}$. What are the constraints on this transformation? If $\beta(E)$ is the original root bound function, this transformation will lead naturally to a corresponding k-ary root bound $\beta_k(E)$. In this paper, our basic goal is to ensure that β_k dominates β:

$$\beta_k(E) \geq \beta(E) \tag{4}$$

for $E \in \mathcal{E}$. Achieving this inequality will depend on the nature of β. Assuming both sides of (3) are well-defined, we have

$$\begin{aligned} E \neq 0 &\Rightarrow E_k \neq 0 \\ &\Rightarrow |E_k| > \beta(E_k) \\ &\Rightarrow |E| > k^{v(E)} \beta(E_k) \end{aligned}$$

Thus we define

$$\beta_k(E) := k^{v(E)} \beta(E_k).$$

and so the inequality (4) amounts to $\beta(k^{v(E)} E_k) \leq k^{v(E)} \beta(E_k)$.

To simplify [3] the presentation below, we will choose $k = 2$. Also, we will simply write $v(E)$ instead of $v_2(E)$. Generalizing this to a general $k > 2$ is mostly straightforward. A further generalization is to maintain the powers of two or more k's simultaneously. It seems that $(k', k'') = (2, 5)$ will yield most of the

[3] The case $k = 2$ is the most important case. Also, the resulting formulas are easier to read as we avoid the use of the variable k.

benefits of the method, since actual input numbers in computation are overwhelmingly decimal or binary. This amounts to the following transformation (cf. (3)):

$$E = 2^{v_2(E)} 5^{v_5(E)} E_{2,5} \tag{5}$$

where $v_k(E) \in \mathbb{Z}$ (for $k = 2, 5$).

3 The BFMSS Bound

We first review the BFMSS Bound [2,3] for algebraic expressions. Let E be an expression as represented by a DAG, with integers at its leaves, and whose internal nodes correspond to the operators in column 1 of Table 2. The "diamond operator" in the last row of the Table extracts the jth largest real root of the polynomial $\sum_{i=0}^{n} F_i X^i$ where F_i are expressions. With the diamond operators of a degree n, we associate an inclusion root bound function (in the sense of [3]),

$$\Phi(a_{n-1}, \ldots, a_i, \ldots, a_0) = \Phi(\ldots, a_i, \ldots) \tag{6}$$

where each a_i will be replaced by F_i/F_n when applied to the polynomial $\sum_{i=0}^{n} F_i X^i$. In the notation $\Phi(\ldots, a_i, \ldots)$, it is understood that the index i decreases from $n-1$ to 0. Since there are several possible choices Φ_1, Φ_2, etc for Φ, we may just compute the bound given by each Φ_i and take the best. This procedure amounts to the observation that if Φ_1 and Φ_2 are exclusion root bound functions, then $\max\{\Phi_1, \Phi_2\}$ is also an exclusion root bound.

The BFMSS bound constructively maintains two real parameters $u(E)$ and $\ell(E)$ as shown in Table 2. To avoid clutter in the table, we write u', u'' for $u(E')$ and $u(E'')$; similarly for ℓ', ℓ''. Furthermore, the diamond operator involves subexpressions $F_0, F_1, \ldots, F_n$; in this case, we write

$$D_i := \frac{u(F_i)}{\ell(F_i)} \prod_{j=0}^{n} \ell(F_j). \tag{7}$$

The **degree** of a node in E is p if the node is the operator $\sqrt[p]{\cdots}$, and n if the node is the diamond operator of degree n. Otherwise the degree is 1. Moreover, let $D(E)$ be the product of all the degrees of the distinct nodes in the dag of E. The degree of $\mathrm{val}(E)$ is bounded by $D(E)$. The BFMSS bound says that

Table 2
BFMSS Rules

E	$u(E)$	$\ell(E)$
integer n	$\lvert n\rvert$	1
$E' \pm E''$	$u'\ell'' + \ell' u''$	$\ell'\ell''$
$E' \times E''$	$u'u''$	$\ell'\ell''$
$E' \div E''$	$u'\ell''$	$\ell' u''$
$\sqrt[p]{E'}$	$\min(\sqrt[p]{u'\ell'^{p-1}}, u')$	$\min(\ell', \sqrt[p]{u'^{p-1}\ell'})$
$\diamond(j, F_n, F_{n-1}, \ldots, F_0)$	$\Phi(\ldots, (D_n)^i D_{n-i}, \ldots)$ where D_i is given in (7)	D_n

if $\mathrm{val}(E) \neq 0$ then

$$|\mathrm{val}(E)| \geq \frac{1}{u(E)^{D(E)-1}\ell(E)}. \tag{8}$$

Hence we may define the BFMSS root bound function as

$$\beta^{\mathrm{bfmss}}(u, \ell, D) := \frac{1}{u^{D-1}\ell}, \tag{9}$$

with the usual convention that we write $\beta^{\mathrm{bfmss}}(E)$ for $\beta(u(E), \ell(E), D(E))$. The BFMSS Rules are given in Table 2. Our rule for $\sqrt[p]{E'}$ in this table is a unification of the two cases in the BFMSS presentation. The advantage of having these two cases was an improvement suggested by Yap.

4 Generalization of BFMSS

Let α be an algebraic number. As in [8], let $\mu(\alpha) = \max\{|\alpha_i| : i = 1, \ldots, n\}$ where $\alpha = \alpha_1, \ldots, \alpha_n$ are all conjugates of α. We call a triple (u', ℓ', v) a **set of $ul[2]$-parameters** for α if $u', \ell' \in \mathbb{R}_{\geq 0}$ and $v \in \mathbb{Z}$ and there exist algebraic integers α_1, α_2 such that

$$\alpha = 2^v \frac{\alpha_1}{\alpha_2}, \tag{10}$$

$\mu(\alpha_1) \leq u'$ and $\mu(\alpha_2) \leq \ell'$. If "2" is replaced by an integer $k > 2$, we have the analogous set of $ul[k]$-parameters. When α is non-zero with degree D, we have

$$|\alpha| \geq \beta_2(u', \ell', v, D) := 2^v \frac{1}{u'^{D-1}\ell'} \tag{11}$$

where $\beta_2(u', \ell', v, D) = \beta_2^{\text{bfmss}}(u', \ell', v, D)$ is the binary version of the BFMSS root bound function. The expression (10) is non-unique. Indeed, there is some leeway for designing a suitable set of $ul[2]$-parameters for α because in general the best choice is not easily given by a fixed rule. Thus, if (u', ℓ', v) is a set of $ul[2]$-parameters for α, then so is either $(u'2^v, \ell', 0)$ or $(u', \ell'2^{-v}, 0)$, depending on whether $v \geq 0$ or not. More generally, it is always possible to reduce $|v|$ towards 0 in any set of parameters (u', ℓ', v). A set of $ul[2]$-parameters is a generalization of the BFMSS parameters, since the BFMSS parameters may be regarded as the special case of $v = 0$.

The BFMSS[2] Rules. The binary transformation of BFMSS is given in Table 3. The table incorporates a refinement of the $ul[2]$-parameters, whereby $v(E)$ is represented by two numbers $v^+(E) \geq 0$ and $v^-(E) \geq 0$ satisfying the relation

$$v(E) = v^+(E) - v^-(E).$$

This refinement will better quantify our gain over the original BFMSS bound (see Lemma 1 below). In actual implementation, it is sufficient to just maintain $v(E)$. In this case, to apply the rules, we will define $v^+(E)$ to be $v(E)$ if $v(E) \geq 0$ and otherwise let $v^+(E) = 0$. Similarly, $v^-(E)$ is defined to be $-v(E)$ if $v(E) < 0$ and otherwise $v^-(E) = 0$. Call this variation the **reduced** version of the BFMSS[2] Rules (in contrast to the **refined** version where v^+, v^- are independent).

When α is represented by an expression E (in the dag form), this table defines a unique set of $ul[2]$-parameters for E,

$$(u_2(E), \ell_2(E), v(E)).$$

The BFMSS[2] root bound for E is

$$E \neq 0 \Rightarrow |E| \geq \frac{2^{v(E)}}{u_2(E)^{D(E)-1}\ell_2(E)} \tag{12}$$

In the table, (u', ℓ', v') denotes the $ul[2]$-parameters of subexpression E'; similarly (u'', ℓ'', v'') is for E''.

Most of the rules in Table 3 can be read off the table; but the more complex diamond operator will be explained here. We want a set of $ul[2]$-parameters for $\diamond(j; F_n, F_{n-1}, \ldots, F_0)$. Suppose $\Phi(a_{n-1}, a_{n-2}, \ldots, a_0)$ is a root bound function in the sense of [3]. Write v_i for $v_i^+ - v_i^- = v^+(F_i) - v_i^-(F_i)$. Define

Table 3
The Refined BFMSS[2] Rules

E	$u_2 = u_2(E)$	$\ell_2 = \ell_2(E)$	$v^+ = v^+(E)$	$v^- = v^-(E)$
binary rational $n2^m$	$\lvert n \rvert$	1	$\max(0, m)$	$\max(0, -m)$
$E' \pm E''$	$2^{v'^+ + v''^- - v^+} u_2' \ell_2'' + 2^{v'^- + v''^+ - v^+} \ell_2' u_2''$	$\ell_2' \ell_2''$	$\min(v'^+ + v''^-, v'^- + v''^+)$	$v'^- + v''^-$
$E' \times E''$	$u_2' u_2''$	$\ell_2' \ell_2''$	$v'^+ + v''^+$	$v'^- + v''^-$
$E' \div E''$	$u_2' \ell_2''$	$\ell_2' u_2''$	$v'^+ + v''^-$	$v'^- + v''^+$
$\sqrt[p]{E'}$, $2^{v'} u_2' \geq \ell_2'$	$\sqrt[p]{2^{\tilde{v} - pv^+} u_2' {\ell_2'}^{p-1}}$	ℓ_2'	$\lceil \widetilde{v}/p \rceil$ where $\widetilde{v} = v'^+ + (p-1)v'^-$	v'^-
$\sqrt[p]{E'}$, $2^{v'} u_2' < \ell_2'$	u_2'	$\sqrt[p]{2^{\tilde{v} - pv^-} {u_2'}^{p-1} \ell_2'}$	v'^+	$\lceil \widetilde{v}/p \rceil$ where $\widetilde{v} = (p-1)v'^+ + v'^-$
$\diamond(j; F_d, , \ldots, F_0)$	$\Phi(\ldots, C_n^i C_{n-i}, \ldots)$ (see (14))	$2^{-w_n} C_n$	0	w_n (see (13))

$$\begin{aligned} w_i &:= v_i + \left(\sum_{j=0}^{n} v_j^- \right) \\ &= v_0^- + \cdots + v_{i-1}^- + v_i^+ + v_{i+1}^- + \cdots + v_n^- \end{aligned} \tag{13}$$

and

$$C_i = 2^{w_i} \frac{u_2(F_i)}{\ell_2(F_i)} \prod_{j=0}^{n} \ell_2(F_j). \tag{14}$$

Just as in BFMSS, the diamond operator (if well-defined) $\diamond(j; F_n, F_{n-1}, \ldots, F_0)$ specifies an algebraic number α where $\alpha = U/L$ and U, L are algebraic integers satisfying

$$\mu(U) \leq \Phi(\ldots, (C_n)^i C_{n-i}, \ldots), \quad \mu(L) \leq C_n.$$

Also, a set of $ul[2]$-parameters for α is

$$(\Phi(\ldots, (C_n)^i C_{n-i}, \ldots), 2^{-w_n} C_n, -w_n). \tag{15}$$

This justifies the rule for diamond operator in Table 3 (Other rules will be justified below).

If we know more about the nature of Φ, improved bounds may be possible. E.g., using the Lagrange-Zassenhaus bound [15], we get the simpler set of $ul[2]$-parameters,

$$(\Phi(\ldots, D_{n-i}, \ldots), 1, 0)$$

where D_{n-i} is given by refeqdi.

BFMSS[2] dominates BFMSS We first prove a key relationship between the BFMSS Rules and the new BFMSS[2] Rules.

LEMMA 1 *Let*

$$(u, \ell), \qquad (u_2, \ell_2, v^+, v^-)$$

be the parameters for an expression E given by Table 2 and Table 3, respectively. Then

$$u = 2^{v^+} u_2, \qquad \ell = 2^{v^-} \ell_2.$$

PROOF. We use induction on the structure of E. The base case is obvious.

CASE $E = E' \pm E''$:

$$\begin{aligned} \frac{u}{\ell} &= \frac{u'\ell''+\ell'u''}{\ell'\ell''} \\ &= \frac{2^{v'^+ + v''^-} u'_2\ell''_2 + 2^{v'^- + v''^+} \ell'_2 u''_2}{2^{v'^- + v''^-} \ell'_2\ell''_2} \\ &= \frac{2^{v^+}(2^{v'^+ + v''^- - v^+} u'_2\ell''_2 + 2^{v'^- + v''^+ - v^+} \ell'_2 u''_2)}{2^{v'^- + v''^-} \ell'_2\ell''_2} \\ &= \frac{2^{v^+} u_2}{2^{v^-}\ell_2} \end{aligned}$$

where $v^+ = \min(v'^+ + v''^-, v'^- + v''^-)$ and $v^- = v'^- + v''^-$. We want to conclude from this derivation that

$$u = 2^{v^+} u_2, \qquad \ell = 2^{v^-} \ell_2.$$

This is only valid if, in the above derivation, we never apply any cancellation of terms between the numerator and denominator. The reader may verify this is the case. In other words, although we presented the argument as a sequence of equations involving ratios, it should be read as a pair of parallel transformations involving the numerator and denominator *separately*. This will also be true in all the other derivations in this proof.

CASE $E = E' \times E''$:

$$\begin{aligned} \frac{u}{\ell} &= \frac{u'u''}{\ell'\ell''} && \text{(BFMSS)} \\ &= \frac{2^{v'^+ + v''^+} u'_2 u''_2}{2^{v'^- + v''^-} \ell'_2\ell''_2} && \text{(induction)} \\ &= \frac{2^{v^+} u_2}{2^{v^-}\ell_2} && \text{(BFMSS[2])} \end{aligned}$$

where $v^+ = v'^+ + v''^+$ and $v^- = v'^- + v''^-$. The division case is similar.

CASE $E = E' \div E''$:

$$\begin{aligned} \frac{u}{\ell} &= \frac{u'\ell''}{\ell'u''} && \text{(BFMSS)} \\ &= \frac{2^{v'^+ + v''^-} u'_2 \ell''_2}{2^{v'^- + v''^+} \ell'_2 u''_2} && \text{(induction)} \\ &= \frac{2^{v^+} u_2}{2^{v^-}\ell_2} && \text{(BFMSS[2])} \end{aligned}$$

where $v^+ = v'^+ + v''^-$ and $v^- = v'^- + v''^+$.

CASE $E = \sqrt[p]{E'}$: The rules here split into two cases, depending on whether $2^v u'_2 \geq \ell'_2$. The critical observation is that $2^v u'_2 \geq \ell'_2$ is equivalent to $u' \geq \ell'$ (the corresponding criteria for choosing between the two cases in the BFMSS Rule). First assume $2^{v'} u'_2 \geq \ell'_2$. Let $\widetilde{v} = v'^+ + (p-1)v'^-$, $v^+ = \lfloor \widetilde{v}/p \rfloor$ and $v^- = v'^-$. We have

$$\begin{aligned} \tfrac{u}{\ell} &= \tfrac{\sqrt[p]{u'\ell'^{p-1}}}{\ell'} && \text{(BFMSS)} \\ &= \tfrac{\sqrt[p]{2^{v'^+ + (p-1)v'^-} u'_2 \ell_2'^{p-1}}}{2^{v'^-}\ell'_2} && \text{(induction)} \\ &= \tfrac{2^{v^+}u_2}{2^{v^-}\ell_2} && \text{(BFMSS[2]).} \end{aligned}$$

The other case, when $2^{v'} u'_2 < \ell'_2$ is similarly shown.

CASE $E = \diamond(F_n, \ldots, F_0)$: For $i = 0, \ldots, n$, we have

$$\begin{aligned} D_i &= \tfrac{u(F_i)}{\ell(F_i)} \textstyle\prod_{j=0}^n \ell(F_j) && \text{(BFMSS)} \\ &= 2^{w_i} \tfrac{u_2(F_i)}{\ell_2(F_i)} \textstyle\prod_{j=0}^n \ell_2(F_j) && \text{(induction)} \\ &= C_i && \text{(BFMSS[2])} \end{aligned}$$

Thus

$$\begin{aligned} u(E) &= \Phi(\ldots, D_n^i D_{n-i}, \ldots) \\ &= \Phi(\ldots, C_n^i C_{n-i}, \ldots) \\ &= u_2(E) = 2^{v^+} u_2(E). \end{aligned}$$

Similarly, $\ell(E) = D_n = C_n = 2^{w_n}\ell_2(E) = 2^{v^-}\ell_2(E)$.

Our main result concerning the BFMSS and BFMSS[2] Rules is the following domination relation:

THEOREM 2 *For expression E supported by Table 2, we have*

$$\beta_2^{\mathrm{bfmss}}(E) \geq \beta^{\mathrm{bfmss}}(E)$$

PROOF. Let $\beta(E) = \frac{1}{u^{D-1}\ell}$ and $\beta_2 = \frac{2^v}{u_2^{D-1}\ell_2}$ be (respectively) the BFMSS and BFMSS[2] bounds for expression E. From Lemma 1, we conclude

$$\frac{\beta_2}{\beta} = \frac{2^v \cdot (2^{v^+}u_2)^{D-1} \cdot (2^{v^-}\ell_2)}{u_2^{D-1}\ell_2} = 2^{v^+ D} \geq 1.$$

Correctness and the Umbral Convention. We now justify the BFMSS[2] Rules in Table 3. The correctness of a set (u_2, ℓ_2, v) of $ul[2]$-parameters for an expression E depends on the existence of algebraic integers U_2, L_2 such that

$$E = 2^v \frac{U_2}{L_2}. \tag{16}$$

with $u_2 \geq \mu(U_2)$, $\ell_2 \geq \mu(L_2)$. We have not given explicit rules for maintaining U_2, L_2, but these are easily deduced from Table 3. That is because the rules for maintaining u_2, ℓ_2 is a "shadow" of the corresponding rules for U_2, L_2. Let us illustrate this: when $E = E' \pm E''$, we have the rule

$$u_2 = 2^{v'^+ + v''^- - v^+} u_2' \ell_2'' + 2^{v'^- + v''^+ - v^+} \ell_2' u_2'' \tag{17}$$

This is a "shadow" of the corresponding[4] rule for U_2:

$$U_2 = 2^{v'^+ + v''^- - v^+} U_2' L_2'' \pm 2^{v'^- + v''^+ - v^+} L_2' U_2'' \tag{18}$$

REMARKS: The original BFMSS rules also have such an umbral connection between (u, ℓ) and the pair of expressions (U, L), although this was only implicit. Such a shadowing technique is similar to the mnemonic device called symbolic or "umbral calculus" from the invariant theorists, and developed by Rota and his collaborators [11] as a form of linear operator.

The umbral relation between (u_2, ℓ_2) and (U_2, L_2) is justified by the following:

Lemma 3 *For any expression E,*
(i) The expressions $U_2(E)$ and $L_2(E)$ are algebraic integers.
(ii) The following inequalities hold:

$$u_2 \geq \mu(U_2), \qquad \ell_2 \geq \mu(L_2). \tag{19}$$

PROOF. (i) We sketch the justification of the rules for $U_2(E)$; the justification of $L_2(E)$ is analogous. Consider the case when $E = E' \pm E''$. Then $U_2(E)$ is given by (18), and this is an algebraic integer because $v'^+ + v''^- - v^+ \geq 0$ and $v'^- + v''^+ - v^+ \geq 0$ (also, inductively, the subexpressions U_2', U_2'' are algebraic integers). In the case of radicals, we use the fact that $\sqrt[p]{E'}$ is an algebraic integer when E' is an algebraic integer. The remaining cases are just as easily

[4] Note that the rules for u_2, ℓ_2 shadow the rules for U_2, L_2, but not vice-versa, because $\pm$ for U_2, L_2 becomes a $+$ for u_2, ℓ_2. This can be seen by comparing (17) and (18).

shown. (ii) We sketch the argument for part (ii). The relationship (19) holds because for algebraic integers A, B, if $a \geq \mu(A)$ and $b \geq \mu(B)$ then

$$a + b \geq \mu(A \pm B), \quad ab \geq \mu(AB), \quad \sqrt[p]{a} \geq \mu(\sqrt[p]{A}).$$

In particular, this justifies why (17) is an upper bound on the algebraic integer (18).

We are ready to prove the correctness of our rules.

THEOREM 4 *Table 3 is correct: for each expression E, the triple $(u_2(E), \ell_2(E), v(E))$ is a set of $ul[2]$-parameters for E.*

PROOF. Since we already know Lemma 3, it remains to show the relation (16). The BFMSS rules produce a pair of algebraic integer expressions $U(E), L(E)$ such that $E = U(E)/L(E)$. Lemma 1 shows that

$$\frac{u}{\ell} = \frac{2^{v^+} u_2}{2^{v^-} \ell_2}.$$

From the umbral relation between (u, ℓ) and (U, L), and also between (u_2, ℓ_2) and (U_2, L_2), we conclude that

$$\frac{U}{L} = \frac{2^{v^+} U_2}{2^{v^-} L_2} = 2^v \frac{U_2}{L_2}.$$

Generalization. We can generalize the $ul[2]$-parameters to $ul[k]$-parameters for any integer $k > 2$. Since the majority of input constants in scientific and engineering computations is covered by the $ul[2]$ or $ul[10]$, the following generalization will be useful: if $q_1, \ldots, q_n \geq 2$ are relatively prime, it is easy to define a set

$$(u(E), \ell(E), v_{q_1}(E), \ldots, v_{q_n}(E))$$

of $ul[q_1, \ldots, q_n]$-parameters for E, so that

$$E = \frac{u(E)}{l(E)} \prod_{i=1}^{n} q_i^{v_{q_i}}$$

Special Cases. The binary BFMSS Rules allow the root bounds of a floating point constant to behave like an integer (i.e., $\ell(E) = 1$). As long as there is no explicit division in our expression, the expression continues to behave like an integer. This is a very important case in practice.

Let us consider some specialization of our rules. Suppose E' and E'' are "almost division-free" in the sense that $\ell_2' = \ell_2'' = 1$ (they may not be algebraic integers since v_1, v_2 can be negative). Then the rule for $E = E' \pm E''$ in Table 3 gives

$$u_2 = 2^{v'^+ + v''^- - v^+} u_2' + 2^{v'^- + v''^+ - v^+} u_2''. \tag{20}$$

When $v' = v''$, this further simplifies to $u_2 = u_2' + u_2''$. Similarly $\ell_2 = 1$ and $v = v'$. Suppose x, y are two L-bit binary numbers. Such numbers can be represented by a binary string of length L with a binary point somewhere in the string. So the triple $(4^L, 1, -L)$ is a set of $ul[2]$-parameters for x and for y. From the preceding, $x+y$ has $ul[2]$-parameters $(2 \cdot 4^L, 1, -L)$. Similarly, xy has the $ul[2]$-parameters $(4^{2L}, 1, -2L)$. Now suppose E is the determinant of an $n \times n$ matrix with entries which are L-bit binary numbers. Viewing E as the standard sum of $n!$ terms, we easily see that E has

$$(4^{nL} n!, 1, -nL) \tag{21}$$

as a set of $ul[2]$-parameters. Furthermore, since $D(E) = 1$, $\beta_2^{\mathrm{bfmss}}(E) = 2^{-nL}$. This justifies the root bit bound given in (2).

5 The k-ary Measure Bound

The Measure Bound Rules from Li-Yap [8] (cf. [10,1]) is shown in the first two columns of table 5. For each expression E, it maintains $\mathrm{M}(E)$ according to the table. The degree bound $D(E)$ is independently computed as usual. If E' is a subexpression, we write M' and D' for $\mathrm{M}(E')$ and $D(E')$; similarly for M'' and D''.

Notice that Line 7 refers[5] to the $\mathrm{Root}(F_n, \ldots, F_0)$ operator. This is just a special case of the diamond operator of BFMSS in which the F_i's are now explicit integers. Thus, such root operators are (essentially) leafs in an expression DAG. This case is not only the most important, it is the version for which efficient algorithms exist. The general diamond operator does not seem easy to implement. Again, we use any inclusion bounding function Φ. The rule here for $v(E)$ refers to the generalized Kronecker symbol $\delta(\cdots)$ where $\delta(X) = 1$ if the predicate X is true, and $\delta(X) = 0$ otherwise. Extracting $v(E)$ from the arguments F_i is relatively straightforward in practice, assuming a binary representation of integers.

[5] Like the diamond operator, one must also specify some parameter to identify a particular root of the polynomial. This extra parameter is omitted here since the root bounds do not depend on it.

Table 4
Binary Measure Bound

	E	$\mathrm{M}(E)$	$v(E)$	$\mathrm{M}_2(E)$
0	Constant 1	1	0	1
1	$2^{v'}\frac{a}{b}E''$ $(v' \in \mathbb{Z})$	$\max\{2^{v'^{+}}\lvert a\rvert,$ $2^{v'^{-}}\lvert b\rvert\}^{D''}\mathrm{M}''$	$v'+v''$	$\max\{\lvert a\rvert,\lvert b\rvert\}^{D''}\mathrm{M}_2''$
2	$E' \times E''$	$\mathrm{M}'^{D''}\mathrm{M}''^{D'}$	$v'+v''$	$\mathrm{M}_2'^{D''}\mathrm{M}_2''^{D'}$
3	$E' \div E''$	$\mathrm{M}'^{D''}\mathrm{M}''^{D'}$	$v'-v''$	$\mathrm{M}_2'^{D''}\mathrm{M}_2''^{D'}$
4	$E' \pm E''$	$\mathrm{M}'^{D''}\mathrm{M}''^{D'}2^{D(E)}$	$v=\mathrm{sign}(v')\min\{\lvert v'\rvert,\lvert v''\rvert\}$ if $v'v''\geq 0$; else $v=0$.	$2^{D+d}\mathrm{M}_2'^{D''}\mathrm{M}_2''^{D'}$, $d=\lvert v'-v''\rvert D'D''$
5	Radical $\sqrt[p]{E'}$	M'	$\lfloor v'/p \rceil$	$2^{\lvert m\rvert D'}\mathrm{M}_2'$, $m=v'-vp$
6	Power E'^k	M'^k	$v'k$	$\mathrm{M}_2'^k$
7	$\mathrm{Root}(F_n,\ldots,F_0)$, $F_i=2^{v_i}a_i$, $v_i \geq 0$ $(0\leq i\leq n)$	$\Phi\left(\frac{F_{n-1}}{F_n},\ldots,\frac{F_0}{F_n}\right)$	$v=-\max\{d: v_i\geq id, 1\leq i\leq n\}$	$\Phi\left(\frac{f_{n-1}}{f_n},\ldots,\frac{f_0}{f_n}\right)$, and $f_i=2^{v_i-iv}a_i$.

The binary version of the measure bound is shown in the last two columns of Table 5. For each expression E, this maintains $v(E)$ and $\mathrm{M}_2(E)$. Note that in Line 5 ($E=\sqrt[p]{E'}$), we choose $v=\lfloor v'/p\rceil$ which is the rounded value of v'/p. For any real r, $\lfloor r\rceil$ is defined to be either $\lfloor r\rfloor$ or $\lceil r\rceil$, subject to $|r-\lfloor r\rceil\,|\leq 1/2$ (we are indifferent as to how a tie is broken). Other choices are possible (e.g., $v=\lfloor v'/p\rfloor$) but the optimal choice cannot be decided by a local rule.

Binary Measure Parameters. Call $(x,y)\in\mathbb{Z}\times\mathbb{R}_{\geq 1}$ a set **binary measure parameters** for an expression E if there exists an expression E_2 such that

$$E=2^x E_2, \qquad \mathrm{M}(E_2)\leq y. \tag{22}$$

LEMMA 5 *If $v(E)$ and $\mathrm{M}_2(E)$ are defined as in Table 5, then $(v(E),\mathrm{M}_2(E))$ is a set of binary measure parameters for E.*

PROOF. We first claim that $E=2^{v(E)}E_2$ for a suitable E_2. We augment Table 5 with another column showing how E_2 is defined inductively; the result is displayed as Table 5.

Most of the entries in the last column are straightforward, so we skip the verification. We only verify two cases for $\mathrm{M}_2(E)$:

Line 4, $E=E'\pm E''$. Here, we have $E_2=2^{v'-v}E_2'+2^{v''-v}E_2''$, and based on the Measure Rules, we can compute $\mathrm{M}(2^{v'-v}E_2')=2^{|v'-v|D'}\mathrm{M}_2'$ and $\mathrm{M}(2^{v''-v}E_2'')=$

Table 5
Correctness of Binary Measure Bound

	E	$v(E)$	$\mathrm{M}_2(E)$	E_2
0	Constant 1	0	1	1
1	$2^{v'}\frac{a}{b}E''$ $(v' \in \mathbb{Z})$	$v' + v''$	$\max\{\lvert a\rvert, \lvert b\rvert\}^{D''}\mathrm{M}_2''$	$\frac{a}{b}E_2''$
2	$E' \times E''$	$v' + v''$	$\mathrm{M}_2'^{D''}\mathrm{M}_2''^{D'}$	$E_2' \times E_2''$
3	$E' \div E''$	$v' - v''$	$\mathrm{M}_2'^{D''}\mathrm{M}_2''^{D'}$	$E_2' \div E_2''$
4	$E' \pm E''$	$v = \mathrm{sign}(v')\min\{\lvert v'\rvert, \lvert v''\rvert\}$ if $v'v'' \geq 0$; else $v = 0$.	$2^{D(E)+d}\mathrm{M}_2'^{D''}\mathrm{M}_2''^{D'}$, $d = \lvert v' - v''\rvert D'D''$	$2^{v'-v}E_2' + 2^{v''-v}E_2''$
5	Radical $\sqrt[p]{E'}$	$\lfloor v'/p \rceil$	$2^{\lvert m\rvert D'}\mathrm{M}_2'$, $m = v' - vp$	$\sqrt[p]{2^m E_2'}$
6	Power E'^k	$v'k$	$\mathrm{M}_2'^k$	E'^k
7	$\mathrm{Root}(F_n, \ldots, F_0)$, $F_i = 2^{v_i}a_i$, $v_i \geq 0$ $(0 \leq i \leq n)$	$v = -\max\{d : v_i \geq id, 1 \leq i \leq n\}$	$\Phi\left(\frac{f_{n-1}}{f_n}, \ldots, \frac{f_0}{f_n}\right)$, and $f_i = 2^{v_i - iv}a_i$.	$\mathrm{Root}(f_n, \ldots, f_0)$

$2^{|v''-v|D''}\mathrm{M}_2''$. Hence

$$\begin{aligned}
\mathrm{M}(2^{v'-v}E_2' + 2^{v''-v}E_2'') &= 2^D\mathrm{M}(2^{v'-v}E_2')^{D''}\mathrm{M}(2^{v''-v}E_2'')^{D'} \\
&= 2^{D+(|v'-v|+|v''-v|)D'D''}\mathrm{M}(E_2')^{D''}\mathrm{M}(E_2'')^{D'} \\
&= 2^{D+|v'-v''|D'D''}\mathrm{M}(E_2')^{D''}\mathrm{M}(E_2'')^{D'} \\
&= 2^{D+d}\mathrm{M}(E_2')^{D''}\mathrm{M}(E_2'')^{D'}.
\end{aligned}$$

This justifies the rule for $M_2(E' \pm E'')$.

Line 7, $E = \mathrm{Root}(F_n, \ldots, F_0)$ where each $F_i = 2^{v_i}a_i$ and $v_i \geq 0$. Let $P(X) = \sum_{i=0}^n F_i X^i$. From $E = 2^v E_2$, we conclude that E_2 is a root of the polynomial $Q(X) = P(2^v X) = \sum_{i=0}^n 2^{v_i + iv}a_i X^i$. But $v = -\max\{d : v_i \geq id, i = 1, \ldots, n\} \geq 0$ implies that $Q(X) = \sum_{i=0}^n 2^{v_i - iv}a_i X^i$ is an integer polynomial. With $f_i = 2^{v_i - iv}a_i$, we see that $\mathrm{M}_2(E) = \mathrm{M}(E_2) = \Phi(f_{n-1}/f_n, \ldots, f_0/f_n)$ is a valid rule.

Domination. Let $\beta(E)$ and $\beta_2(E)$ be the rootbounds function associated with the original measure bound and the binary measure bound. Indeed,

$$\beta(E) = \frac{1}{\mathrm{M}(E)}, \qquad \beta_2(E) = \frac{2^{v(E)}}{\mathrm{M}_2(E)}. \tag{23}$$

It is easy to see that $\beta_2(E)$ is a rootbound for E since $E \neq 0$ implies

$$\begin{aligned}
|E| &= 2^{v(E)}|E_2| && \text{(by validity of } \beta_2\text{)}\\
&\geq 2^{v(E)}/\mathrm{M}(E_2) && \text{(the usual measure bound)}\\
&\geq 2^{v(E)}/\mathrm{M}_2(E) && \text{(since } \mathrm{M}(E_2) \leq \mathrm{M}_2(E)\text{, putting } y = M_2(E) \text{ in (22))}\\
&= \beta_2(E).
\end{aligned}$$

Our goal is to prove that β_2 dominates β in the sense that $\beta_2(E) \geq \beta(E)$ for all E supported by the operators in Table 5. Unfortunately, we cannot do this without further information about the root bound function $\Phi(\cdots)$. We get around this problem by modifying Line 7 in 5 so that $v(E) = 0$ and $M_2(E) = M(E)$. Call this the "trivial rule" for the root operator. We prove a basic inequality:

LEMMA 6 *Let E be any expression involving the operators of Table 5. Assuming the trivial rule for the root operator, we have*

$$\mathrm{M}(E) \geq 2^{|v(E)|D(E)}\mathrm{M}_2(E).$$

PROOF. We verify this lemma for each line of the Table 5.
Line 0. When $E = 1$, the lemma is immediate since $v(E) = 0$ and $\mathrm{M}(E) = \mathrm{M}_2(E)$.
Line 1. Actually, Line 1 is a special case of Line 2, with $E' = 2^{v'}a/b$ (so $D' = 1$). So we just have to check that $M_2' = \mathrm{M}(a/b) = \max\{|a|, |b|\}$.
Line 2, $E = E'E''$.

$$\begin{aligned}
\mathrm{M}(E) &= \mathrm{M}'^{D''}\mathrm{M}''^{D'} && \text{(Measure Rules)}\\
&\geq 2^{(|v'|+|v''|)D'D''}\mathrm{M}_2'^{D''}\mathrm{M}_2''^{D'} && \text{(by induction)}\\
&\geq 2^{|v'+v''|D}\mathrm{M}_2'^{D''}\mathrm{M}_2''^{D'} && (|v'| + |v''| \geq |v' + v''|,\ D'D'' \geq D)\\
&= 2^{|v|D}\mathrm{M}_2(E) && \text{(Measure[2] Rules)}
\end{aligned}$$

Line 3, $E = E'/E''$. This is similar to the proof of Line 2, since the usual Measure Rules for division is exactly the same as for multiplication. However, in the Measure[2] Rules, $v = v' - v''$. This changes one justification in the preceding proof, replacing $|v'|+|v''| \geq |v'+v''| = |v|$ by $|v'|+|v''| \geq |v'-v''| = |v|$.

Line 4, $E = E' \pm E''$.

$$\begin{aligned}
\mathrm{M}(E) &= 2^D \mathrm{M}'^{D''} \mathrm{M}''^{D'} && \text{(Measure Rules)} \\
&\geq 2^{D+(|v'|+|v''|)D'D''} \mathrm{M}_2'^{D''} \mathrm{M}_2''^{D'} && \text{(by induction)} \\
&\geq 2^{D+(|v|+|v'-v''|)D'D''} \mathrm{M}_2'^{D''} \mathrm{M}_2''^{D'} && (|v'|+|v''| \geq |v|+|v'-v''|, \text{ valid even for } v=0) \\
&\geq 2^{|v|D} 2^{D+|v'-v''|D'D''} \mathrm{M}_2'^{D''} \mathrm{M}_2''^{D'} && (D'D'' \geq D) \\
&= 2^{|v|D} \mathrm{M}_2(E) && \text{(Measure[2] Rules)}
\end{aligned}$$

Line 5, $E = \sqrt[p]{E'}$.

$$\begin{aligned}
\mathrm{M}(E) &= \mathrm{M}' && \text{(Measure Rules)} \\
&\geq 2^{|v'|D'} M_2' && \text{(by induction)} \\
&\geq 2^{(|v|+|m|)D'} M_2' && (|v'| \geq |v|+|m|) \\
&= 2^{|v|D} \mathrm{M}_2(E) && \text{(Measure[2] Rules)}
\end{aligned}$$

To see that $|v'| \geq |v| + |m|$, note that the result is immediate if $v = \lfloor v'/p \rceil = 0$ or $m = 0$. Otherwise, the inequality follows from $|v'| \geq (\frac{|v'|}{p} + \frac{1}{2}) + \frac{p}{2}$, i.e., $\frac{|v'|(p-1)}{p} \geq \frac{p+1}{2}$. The latter is true since $|v'| \geq p+1$ and $p \geq 2$.
Line 6, $E = E'^k$.

$$\begin{aligned}
\mathrm{M}(E) &= \mathrm{M}'^k && \text{(Measure Rules)} \\
&\geq 2^{|v'|kD'} M_2'^k && \text{(by induction)} \\
&= 2^{|v|D} \mathrm{M}_2(E) && \text{(Measure[2] Rules)}
\end{aligned}$$

Line 7, $E = \mathrm{Root}(F_n, \ldots, F_0)$. If we adopt the trivial rule where $v(E) = 0$ then the inequality of this lemma is also trivial.

The main domination result is now easy to show:

THEOREM 7 *Let E be any expression involving the operators of Table 5. Assuming the trivial rule for the root operator, we have*

$$\beta(E) \leq \beta_2(E).$$

PROOF. We have

$$\begin{aligned}\beta(E) &= \tfrac{1}{\mathrm{M}(E)} && \text{(by definition)}\\ &\le \tfrac{1}{\mathrm{M}_2(E)2^{|v(E)|D}} && \text{(preceding Lemma)}\\ &\le \tfrac{2^{v(E)}}{\mathrm{M}_2(E)} && (|v(E)|D + v(E) \ge 0)\\ &= \beta_2(E)\end{aligned}$$

6 Experimental Results

The timings in this paper are based on runs on an Ultrasparc 10 machine with a 440 MHz CPU. The software is `Core Library` Version 1.5+, which implements [6] the Measure Bound, the Li-Yap Bound and a choice between the original BFMSS, the BFMSS[2], or the BFMSS[2,5] Bound.

To give empirical data on the relative effectiveness of these three bounding functions, we run the `Core Library` Test Suite and counted the number of times that each bounding function is the best one. The results are shown in Table 6. Note that more than one bounding function may be best for an expression. So for each bounding function β, we give a pair m/n of numbers where m is the number of times that β achieves the best bound, and n is the number of times β is the *unique* best bound. Thus $m \ge n$. The first column gives the result of a run with the original BFMSS Bound used, the second column gives the result of a run with the BFMSS[2] Bound used, and the third column gives the result of a run with the BFMSS[2,5] Bound used. We conclude from this table that the $(2,5)$-ary version of BFMSS, for all practical purposes, dominates the other two bounding functions in our test suite.

Experiment 1 involves the expression $E_1(x,y)$ given in the introduction. We assume that $E_1(x,y)$ does not share subexpressions. For example, we can reduce the degree from 16 to 8 by sharing, and the bit-bound function for BFMSS improves to $48L + 22$.

Experiment 2 involves the expression $E_2(x,y) = \frac{\sqrt{x}-\sqrt{y}}{x-y} - \frac{\sqrt{x}-\sqrt{y}}{x-y}$, an example from [3]. When x, y are integers, the bit-bound from BFMSS and Li-Yap are $6L + 64$ and $65L + 91$, respectively. But when x, y are L-bit binary numbers, the bit-bound of BFMSS[2] is $7.5L + 11$. When we substitute various machine

[6] Version 1.5+ refers to the modifications of the released Version 1.5 necessary to support the experiments of this paper. Our implementation of these bounds will generally be slightly worse bound than the theory predicts because we maintain upper bounds on $\lg \mathrm{M}(E), \lg u_2(E)$, etc, instead $\mathrm{M}(E), u_2(E)$, etc. The `Core Library` Test Suite is a set of about 30 sample programs that is distributed with the library.

Table 6
Relative effectiveness of 3 Root Bounds on CORE Test Suite

	original BFMSS	BFMSS[2]	BFMSS[2,5]
BFMSS family	55712/4016	55726/14214	55746/15277
Li-Yap	51669/33	41531/19	40472/3
degree-measure	4/4	4/4	0/0
Total number of expressions	55749	55749	55749

double values, we obtain bit-bounds whose ranges are: 1643-1707 (BFMSS), 323-331 (BFMSS[2]). Running these 1000 times gives timings of 36 seconds (BFMSS) and 22.8 seconds (BFMSS[2]). Although there is an improvement, it is not of the order of magnitude one might expect from bit-bound ranges; this seems to be an implementation-induced effect.

Determinants. Experiment 3 involves the determinant example in the introduction. Let A be a $n \times n$ matrix whose entries are **L-bit binary rationals**. By definition, the entries have the form $n2^{-k}$ where $0 \leq n < 2^L$ and $0 \leq k \leq L$. There are two special cases that we consider:
(1) If $n \geq 2^{L-1}$, we say the L-bit binary rational is **strict**. All the numbers in A are strict in our experiment.
(2) If $k = L$, then we say the L-bit binary rational is **normal**.

We noted that if E_0 is the co-factor expansion of matrix A, then the BFMSS bound gives $-\lg \beta^{\rm bfmss}(E_0) \leq (n!)nL$, while the binary BFMSS bound gives $-\lg \beta_2^{\rm bfmss}(E) \leq nL$. If E' is the dynamic programming implementation of the determinant of A, then $\beta^{\rm bfmss}(E')$ may be strictly greater than $\beta(E)$. For instance, if a, b, c are L-bit binary numbers then $\beta^{\rm bfmss}(a(b+c)) = 3L$ while $\beta^{\rm bfmss}(ab+ac) = 4L$. On the other hand, $\beta_2^{\rm bfmss}(a(b+c)) = \beta_2^{\rm bfmss}(ab+ac)$. Table 7 compares the root bit bounds of BFMSS and the binary version on random matrices whose entries are 100-bit binary rationals. These empirical bounds are (as expected) better than the worst case estimate. If we use normal 100-bit binary rationals, the Table 8 gives the same comparison when those entries are *normalized* 100-bit binary rationals. Our implementation of the $\beta_2^{\rm bfmss}$ bound practically matches the theoretical upper bound of nL.

We next compare timing for BFMSS, BFMSS[2] and BFMSS[2,5]. Despite the wide gap in the root bounds, the timings is not expected to be different for random matrices. That is because a random determinant is unlikely to be zero and so the floating point filter will be in effect. Instead, we convert the above data into degenerate matrices, just by making the last row a duplicate of the previous row. Surprisingly, there was no detectable difference in timing between BFMSS and BFMSS[2]. This could be explained as follows: the internal representation of the numbers was in binary, and even when the root bound

Table 7
Bitbound for dynamic programming determinant for random binary entries ($L = 100$)

n	$(n!)nL$	BFMSS	nL	BFMSS[2]
2	400	164	200	101
3	1800	657	300	169
4	9600	2267	400	248
5	$60,000$	$10,468$	500	326

Table 8
Bitbound for dynamic programming determinant for random *normal* binary entries ($L = 100$)

n	$(n!)nL$	BFMSS	nL	BFMSS[2]
2	400	400	200	200
3	1800	1497	300	300
4	9600	6364	400	400
5	$60,000$	$32,282$	500	499

Table 9
Dynamic programming determinant for degenerate *strict* matrices with 50-digit decimal rationals

n	BFMSS		BFMSS[2]		BFMSS[2,5]	
	time	bitbd	time	bitbd	time	bitbd
2	4.4	352	4.3	300	4.4	175
3	15.2	648	13.9	538	14.1	236
4	57.5	2624	58	1984	35	339
5	568	15,427	572	12,202	107	444

asks for many bits of precision, our implementation of BigFloat ensures that no redundant bits are transmitted (i.e., trailing zeroes are omitted). Hence the speedup could only be observed if we use inputs that are not purely binary. Therefore, in our next set of experiments for timing, we use decimal rationals. We use random matrices whose entries are strict 50-digit decimal rationals. Table 9 compares the speed of BFMSS, BFMSS[2] and BFMSS[2,5]. The timing are for $10,000$ evaluations of each determinant.

7 Open Problems and Future Work

This paper introduced the factoring technique into constructive root bounds, and demonstrated its effectiveness. In general, the problem of constructive root bounds will become more important as EGC techniques and such algorithms become more widely used. The trade-offs between effectiveness (i.e., small root-bit bounds) and efficiency (i.e., low computational complexity) is not understood. Between the extremes of simple recursive rules (as constitute the bulk of current bounds) and (say) computing minimal polynomials, we would like to see methods with intermediate computational complexity. Our factoring method can be seen as one step in this direction. We list some open problems and future work:

- Our k-ary method can be generalized to maintain arbitrary rational factors, in addition to k-ary factors. (e.g., transform E to $q2^vE_2$ where $v \in \mathbb{Z}$, $q \in \mathbb{Q}$). The benefit of the rational factors is less predictable, and hence experimentation is called for.
- Current constructive root bound techniques are mostly static in nature. More dynamic root bound techniques should be exploited. An idea of Sekigawa[14] can be pursued. Sekigawa proposed some methods in the case of the measure bound, but they do not seem to have been implemented. We could combine with the most significant bit (MSB) bound that is maintained in the `Core Library` [7].
- It is clear that the k-ary method can also be applied to the Li-Yap Bound.
- The general treatment of the diamond operators under the Measure Bound is subject for further research.
- The incorporation of the Sekigawa improvements into the current Measure[2] Rules is immediate if there is no division. It is possible to give rules that incorporate these improvements for division, but it is unclear how to ensure that the binary bound dominates the original bound.

7.0.0.1 Postscript. Recently, Susanne Schmidt [13] has extended the techniques of this paper for the diamond operator $\diamond(F_n, \ldots, F_0)$, and validated the theoretical improvements by experiments.

References

[1] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra. Exact geometric computation made easy. In *Proc. 15th ACM Symp. Comp. Geom.*, pages 341–450, New York, 1999. ACM Press.

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

[3] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A separation bound for real algebraic expressions. In *Lecture Notes in Computer Science*, pages 254–265. Springer, 2001. to appear, Algorithmica.

[4] Core Library homepage, Since 1999. Software downloads, documentation and links: `http://cs.nyu.edu/exact/core/`.

[5] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A Core library for robust numerical and geometric libraries. In *15th ACM Symp. Computational Geometry*, pages 351–359, 1999.

[6] LEDA Homepage, Since 1994. URL `http://www.mpi-sb.mpg.de/LEDA/`. Library of Efficient Data Structures and Algorithms (LEDA) Project. From the Max Planck Institute of Computer Science.

[7] C. Li. *Exact Geometric Computation: Theory and Applications*. Ph.d. thesis, New York University, Department of Computer Science, Courant Institute, Jan. 2001. Download from `http://cs.nyu.edu/exact/doc/`.

[8] C. Li and C. Yap. A new constructive root bound for algebraic expressions. In *12th ACM-SIAM Symp. on Discrete Algorithms*, pages 496–505, Jan. 2001.

[9] M. Marden. *The Geometry of Zeros of a Polynomial in a Complex Variable.* Math. Surveys. American Math. Soc., New York, 1949.

[10] M. Mignotte. Identification of algebraic numbers. *J. of Algorithms*, 3:197–204, 1982.

[11] G.-C. Rota. *Finite Operator Calculus.* Academic Press, Inc, 1975.

[12] E. R. Scheinerman. When close enough is close enough. *Amer. Math. Monthly*, 107:489–499, 2000.

[13] S. Schmitt. Improved separation bounds for the diamond operator. Technical Report ECG-TR-363108-01, MPI Saarbrücken, 2004.

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

[15] C. K. Yap. *Fundamental Problems of Algorithmic Algebra.* Oxford University Press, 2000.

[16] C. K. Yap. On guaranteed accuracy computation. In F. Chen and D. Wang, editors, *Geometric Computation*. World Scientific Publishing Co., Singapore, 2004.

[17] C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, pages 452–486. World Scientific Press, Singapore, 2nd edition, 1995.