Lecture 12
CONSTRUCTIVE ZERO BOUNDS

This chapter describes some effective methods for computing lower bound on algebraic expressions. Two particular methods will be developed in detail: the measure bound and BFMSS bound.

§1. An Approach to Algebraic Computation

Algebraic numbers that arise in practice are represented by expressions such as $\sqrt{2}+\sqrt{3}-\sqrt{5+2\sqrt{6}}$ or $1-100\sin(1/100)$. The critical question is to determine the sign of such expressions, or to detect when they are undefined. Assume we can approximate any well-defined expression e to any desired absolute precision, i.e., for all $p \in \mathbb{N}$, we can compute an approximate value $\widetilde{e}$ such that $|e-\widetilde{e}| \leq 2^{-p}$. If $e \neq 0$, then we can compute $\widetilde{e}$ for $p = 1, 2, 3, \ldots$ until $|\widetilde{e}| > 2^{-p}$. At this point, we know the sign of e is that of $\widetilde{e}$. The problem is that when $e = 0$, this iteration cannot halt. But suppose we can compute some bound $B(e) > 0$ with the property that if $e \neq 0$ then $|e| > B(e)$. In this case, we can halt our iteration when $p \geq 1 - \lg B(e)$, and declare that $e = 0$. In proof,

$$|e| \leq |\widetilde{e} + 2^{-p} \leq 2^{1-p} \leq B(e)$$

implies $e = 0$.

In general, an **expression** is a rooted DAG over some set Ω of real algebraic operators. A typical set is $\Omega = \{\pm, \times, \div\} \cup \mathbb{Z}$. Note that constants such as $n \in \mathbb{Z}$ are regarded as 0-ary operators, and these appear at the leaves of the DAGs. Let $Expr(\Omega)$ denote the set of expressions over Ω. There is a natural evaluation function $\texttt{val} : Expr(\Omega) \to \mathbb{R}$ such that $\texttt{val}(e)$ is the value denoted by e. In general, $\texttt{val}$ is a partial function, since some operators in Ω (like $\div$) may be partial. We write $\texttt{val}(e) = \uparrow$ in case $\texttt{val}(e)$ is undefined; otherwise we write $\texttt{val}(e) = \downarrow$. A function $B : Expr(\Omega) \to \mathbb{R}_{\geq 0}$ is called a **zero bound function** if for all $e \in Expr(\Omega)$, if $\texttt{val}(e) = \downarrow$ then $|\texttt{val}(e)| \geq B(e)$. Since the lower bound is only in effect when $\texttt{val}(e) = \downarrow$, we may call $B(e)$ a "conditional" lower bound.

We generalize the above observations to a general computational paradigm. This is basically the method encoded in the Core Library. Each algebraic operation in $\Omega = \{+, -, \times, \div, \sqrt{\cdot}, \ldots\}$ is regarded as the construction of a root of a DAG whose leaves are (say) integers. Thus, each node u of the DAG has an implicit real value $\texttt{val}(u)$ (which may be undefined). Moreover, assume that we store two quantities at every node u of the DAG: a precision parameter $p_u \in \mathbb{N}$ and a bigfloat interval $I_u \in \mathbb{Z}[\frac{1}{2}]$ (possibly I_u is undefined). Inductively assume that $\texttt{val}(u) \in I_u$ and $w(I_u) \leq 2^{-p_u}$. Moreover, we assume algorithms which can approximate each operation in Ω to whatever precision we wish. Suppose we want to approximate a given expression e to some absolute precision p. Assume $e = e' \diamond e''$ where $\diamond \in \Omega$. The lazy approach says that we just compute (using interval arithmetic) the value $I_e := I_{e'} \diamond I_{e''}$ and see if $w(I_e) \leq 2^{-p_e}$. If not, we refind the intervals $I_{e'}$ and $I_{e''}$ and repeat. But in Core Library, we do this iteration more actively, by computing the precision $p_{e'}, p_{e''}$ in e' and e'' that will ensure that I_e has precision p_e. This is called "precision-driven computation". All this computation is relatively straightforward in interval arithmetic. What makes our system unique is that we also compute a zero bound $B(e)$ for each expression, and this allows us to decide the sign of e. When $w(I_e) \leq B(e)/2$, and $0 \in I_e$, we conclude that e is zero.

We now focus on how to compute such $B(e)$ bounds. Using the theory of resultants, we can define a suitable zero bound function for expressions over $\Omega = \{\pm, \times, \div, \sqrt{\cdot}\} \cup \mathbb{Z}$. For instance, if $e = e_1 e_2$, then we know from resultants that a defining polynomial $A(X)$ for e can be obtained from the definition polynomials $A_i(X)$'s for e_i $(i = 1, 2)$. Moreover, if $\mathrm{ht(e_i) = h_i}$ then

$$\mathrm{ht(e) \leq h := h_1^{m_2} h_2^{m_1}}$$

where $m_i = \deg(A_i)$. From Cauchy's bound (see previous Chapter), it is clear that we can define $B(e) = (1+h)^{-1}$. There are similar relations for $e = e_1 \pm e_2$, $e = e_1/e_2$, $e = \sqrt[k]{e_1}$. Thus, if we maintain recursively,

for each node in e, an upper bound on the height and degree of the corresponding algebraic number, we can recursive compute $B(e)$ for any expression e. This is the **degree-height bound**, implemented in the first system for such kind of numerical algebraic computation, `Real/Expr` [16] (cf. [15, p. 177]).

In this chapter, we describe several other zero bound functions in detail. The general feature of such **constructive bounds** is illustrated by the degree-height bound. Two ingredients are needed: first, we need a set of recursive rules to maintain a set

$$p_1(e), \ldots, p_t(e)$$

of numerical parameters for an expression e. In the degree-height bound, $t = 2$ where $p_1(e)$ is a height bound and $p_2(e)$ is the degree bound. The second ingredient is a zero bound function $B(e)$ that is computed obtained as a function β of these parameters, $B(e) = \beta(p_1(e), \ldots, p_t(e))$. In the degree-height bound, $\beta(p_1, p_2) = (1 + p_1)^{-1}$.

Such recursive rules apply to a suitable set $Expr(\Omega)$ of expressions. For these notes, we mainly focus on the following set

$$\Omega_2 := \{\pm, \times, \div, \sqrt{\cdot}\} \cup \mathbb{Z}.$$

We can slightly extend Ω_2 to allow $\sqrt[k]{\cdot}$ for integers $k \geq 2$.

Let $B(e) = \beta(p_1(e), \ldots, p_t(e))$ be a constructive zero bound over a class K of expressions. Suppose we have another constructive zero bound $C(e) = \gamma(q_1(e), \ldots, q_u(e))$ over K which is based on a different set of parameters $q_1(e), \ldots, q_u(e)$ and bounding function $\gamma(q_1, \ldots, q_u)$. We would like to compare $C(e)$ and $B(e)$: we say B is as **efficient** than C if for every $e \in K$, the time complexity of maintaining the parameters $p_1(e), \ldots, p_t(e)$ and computing the function $b(p_1, \ldots, p_t)$ is order of the corresponding complexity for C. But there is another way to compare $B(e)$ and $C(e)$: we say $B(e)$ **dominates** $C(e)$ if for all $e \in K$, $B(e) \geq C(e)$. For instance, the degree-measure bound, to be described, dominates the degree-height bound. At least for known constructive bounds, the efficiency issue is less important than having as large a lower bound as possible. Hence we mainly compare zero bounds based on their domination relationship.

Degree Bound for Expressions. In all the zero bounds, we need to maintain an upper bound $D(e)$ on the degree of $\texttt{val}(e)$. Consider an expression e having r radical nodes with indices $k_1, k_2, \ldots, k_r$. Then we claim that the degree of e is at most

$$D(e) = \prod_{i=1}^{r} k_i. \tag{1}$$

In proof, suppose we topologically sort the nodes in e, beginning with a leaf node and ending in the node e. Let the sorted list be $(v_1, v_2, \ldots, v_s)$. Inductively, define $d_1, \ldots, d_s$ where $d_1 = 1$ and d_{i+1} is equal to kd_i if v_{i+1} is k-th root, and otherwise $d_{i+1} = d_i$. It is clear that $d_s = D(e)$. Now it is clear that, by induction, $\deg(v_i) \leq d_i$. This proves our claim. It is also not hard to compute $D(e)$. This method of bounding degree extends to the "RootOf" operator (below) which introduces arbitrary real algebraic numbers.

§1.1. The Mahler Measure bound

Every algebraic number α has a unique minimal polynomial $\mathrm{Irr}(\alpha) \in \mathbb{Z}[X]$. We can factor $\mathrm{Irr}(\alpha)$ over $\mathbb{C}$ as $\mathrm{Irr}(\alpha) = a\prod_{i=1}^{m}(X - \alpha_i)$ with a a positive integer. We may assume $\alpha = \alpha_1$; each α_i is called a **conjugate** of α. Mahler's **measure** of α is defined as

$$M(\alpha) := a \cdot \prod_{i=1}^{m} \max\{1, |\alpha_i|\}.$$

For instance, $M(\sqrt{2}) = 2$ because the minimal polynomial of $\sqrt{2}$ is $X^2 - 2 = (X - \sqrt{2})(X + \sqrt{2})$ and $a = 1$. On the other hand, $M(1 + \sqrt{2}) = 1 + \sqrt{2}$ which is irrational.

In the following, it is convenient to define for any complex z,

$$\max_1(z) := \max\{1, |z|\}. \tag{2}$$

In general, if $A(X) = \sum_{i=0}^{m} a_i X^i \in \mathbb{C}[X]$ is any polynomial, we define its measure as

$$M(A) := |a_m| \cdot \prod_{i=1}^{m} \max_1(\alpha_i),$$

where α_i's are the complex roots of $A(X)$. This definition might appear unnatural but in fact $M(\alpha)$ can also be defined by a natural integral (Exercise). In case $A(X) \in \mathbb{Z}[X]$, we see that $M(A) \geq 1$.

In particular, $M(\alpha) \geq 1$ for non-zero algebraic α. It is also easy to see that if $A(X), B(X) \in \mathbb{Z}[X]$ then

$$M(AB) = M(A)M(B) \tag{3}$$

and hence we conclude that

$$M(A) \leq M(AB).$$

The basis for the degree-measure bound is the following theorem:

Lemma 1.
(i) $|\alpha| \leq M(\alpha)$.
(ii) If $\alpha \neq 0$ *then* $M(1/\alpha) = M(\alpha)$.
(iii) $|\alpha| \geq \frac{1}{M(\alpha)}$.

Proof. (i) is immediate from the definition of measure. For (ii), we observe that if the minimal polynomial of α is $A(X) = \sum_{i=0}^{m} a_i X^i = a_m \prod_{i=1}^{m}(X - \alpha_i)$ then $a_0 \neq 0$ and

$$B(X) = X^m A(1/X) = a_0 \prod_{i=1}^{m}(X - 1/\alpha_i)$$

is (up to sign) the minimal polynomial of $1/\alpha$. Further,

$$a_0 = a_m \prod_{i=1}^{m} \alpha_i \tag{4}$$

Hence

$$\begin{aligned} M(1/\alpha) &= |a_0| \textstyle\prod_i \max_1(1/\alpha_i) \\ &= |a_m| \textstyle\prod_i |\alpha_i| \cdot \max_1(1/\alpha_i) \quad (\text{ by } (4)) \\ &= |a_m| \textstyle\prod_i \max_1(\alpha_i) \\ &= M(\alpha). \end{aligned}$$

Finally, (iii) follows from (i) and (ii): $1/|\alpha| \leq M(1/\alpha) = M(\alpha)$. **Q.E.D.**

In other words, if e is any expression, and we maintain an upper bound $m(e)$ on it's Mahler measure, then the $B(e) \geq 1/m(e)$. Let us now see how we can maintain such an upper bound. As in the case of height, we see that we need to also maintain an upper bound on the degree of e. Such bounds were first exploited by Mignotte [9] in the problem of identification of algebraic numbers.

In the following, we use the following elementary relations:

$$\max_1(\alpha\beta) \leq \max_1(\alpha)\max_1(\beta), \tag{5}$$

$$\max_1(\alpha \pm \beta) \leq 2\max_1(\alpha)\max_1(\beta). \tag{6}$$

The first inequality (5) is trivial in case $|\alpha\beta| \leq 1$. Otherwise,

$$\max_1(\alpha\beta) = |\alpha| \cdot |\beta| \leq \max_1(\alpha)\max_1(\beta).$$

The second inequality is trivial in case $|\alpha \pm \beta| \leq 2$. Otherwise, let $|\beta| \geq |\alpha|$ and

$$\max_1(\alpha \pm \beta) \leq |\alpha| + |\beta| \leq 2|\beta| \leq 2\max_1(\alpha)\max_1(\beta).$$

We have the following relations on measures [10]:

Lemma 2. *Let α and β be two nonzero algebraic numbers of degrees m and n respectively. Let $k \geq 1$ be an integer.*

$$\begin{array}{lllll}
(o) & M(p/q) & \leq & \max|p|,|q|, & p,q \in \mathbb{Z}, q \neq 0 \\
(i) & M(\alpha \times \beta) & \leq & M(\alpha)^n M(\beta)^m & \\
(ii) & M(\alpha/\beta) & \leq & M(\alpha)^n M(\beta)^m & \\
(iii) & M(\alpha \pm \beta) & \leq & 2^d M(\alpha)^n M(\beta)^m, & d = \deg(\alpha \pm \beta) \\
(iv) & M(\alpha^{1/k}) & \leq & M(\alpha) & \\
(v) & M(\alpha^k) & \leq & M(\alpha)^k &
\end{array}$$

Proof. For (o), it sufficient to assume p, q are relatively prime so that the minimal polynomial of p/q is $qX - p$. Then $M(()p/q) = |q| \max_1(p/q)$, which we can verify is equal to $\max|p|,|q|$.

Let the minimal polynomials of α, β be $A(X) = a\prod_{i=0}^{m}(X-\alpha_i)$ and $B(X) = b\prod_{j=0}^{m}(X-\alpha_j)$, respectively. For (i), we use the fact that the minimal polynomial of $\alpha\beta$ divides $a^n b^m \prod_i \prod_j (X - \alpha_i\beta_j)$ which we saw is the resultant $\mathtt{res}_Y(A(Y), Y^n B(X/Y))$. Hence, by (3),

$$\begin{array}{rcl}
M(\alpha\beta) & \leq & a^n b^m \prod_i^m \prod_j^n \max_1(\alpha_i\beta_j) \\
& \leq & a^n \prod_i b \prod_j \max_1(\alpha_i)\max_1(\beta_j) \\
& = & a^n \prod_i \max_1(\alpha_i)^n \left[b \prod_j \max_1(\beta_j)\right] \\
& = & a^n \prod_i \max_1(\alpha_i)^n M(\beta) \\
& = & M(\beta)^m a^n \prod_i \max_1(\alpha_i)^n \\
& = & M(\beta)^m M(\alpha)^n.
\end{array}$$

Part (ii) follows from (i), using the fact that $M(1/\beta) = M(\beta)$ and $\deg(1/\beta) = n$.

For (iii), we use the fact that the minimal polynomial of $\alpha \pm \beta$ divides $a^n b^m \prod_i \prod_j (X - \alpha_i \mp \beta_j)$ which we saw is the resultant $\mathtt{res}_Y(A(Y), B(X \mp Y))$. Let $I \subseteq \{1,\ldots,m\} \times \{1,\ldots,n\}$ such that the conjugates of $\alpha \pm \beta$ is given by the set $\{\alpha_i \pm \beta_j : (i,j) \in I\}$. So $\deg(\alpha \pm \beta) = |I|$.

$$\begin{array}{rcl}
M(\alpha \pm \beta) & \leq & a^n b^m \prod_{(i,j)\in I} \max_1(\alpha_i \pm \beta_j) \\
& \leq & a^n b^m \prod_{(i,j)\in I} 2\max_1(\alpha_i)\max_1(\beta_j) \\
& = & 2^{|I|} a^n b^m \prod_{(i,j)\in I} \max_1(\alpha_i)\max_1(\beta_j) \\
& \leq & 2^{|I|} a^n b^m \prod_i \prod_j \max_1(\alpha_i)\max_1(\beta_j).
\end{array}$$

The rest of the derivation follows as in part (i).

For (iv) and (v), use the facts that the minimal polynomials of $\alpha^{1/k}$ and α^k (respectively) divide $A(X^k) = a\prod_i (X^k - \alpha_i)$ and $A(X)^k$. **Q.E.D.**

Using this lemma, we give a recursive definition of a function $m(e)$, as shown in Table 1 under the column with the heading "$m(e)$". Thus is an upper bound on measure

$$M(e) \leq m(e).$$

Hence we conclude from from Lemma 1 that the function

$$B(e) = 1/m(e)$$

is a root bound function.

Sekigawa [13] gave a refinement of these rules in the case an expression e is division-free. Let $M_0(A)$ denote the leading coefficient of A, and define $M_1(A)$ by the equation

$$M(A) = M_0(A)M_1(A).$$

In case e is an expression and A is the minimal polynomial of $\mathtt{val}(e)$, we write $M_0(e)$ and $M_1(e)$ for $M_0(A)$ and $M_1(A)$ (resp.). Sekigawa gave recursive definitions of two functions $m_0(e)$ and $m_1(e)$ which are upper

	e	$m(e)$	$m_1(e)$	$m_0(e)$
1.	rational p/q	$\max\{\|p\|,\|q\|\}$	$\max_1(p/q)$	$\|q\|$
2.	$e_1 \pm e_2$	$2^{d(e)}m(e_1)^n m(e_2)^m$	$m_1(e_1)^n m_1(e_2)^m$	$2^{d(e)}m_0(e_1)^n m_0(e_2)^m$
3.	$e_1 \times e_2$	$m(e_1)^n m(e_2)^m$	$m_1(e_1)^n m_1(e_2)^m$	$m_0(e_1)^n m_0(e_2)^m$
4.	$e_1 \div e_2$	$m(e_1)^n m(e_2)^m$	—	—
5.	$\sqrt[k]{e_1}$	$m(e_1)$	$m_1(e_1)$	$m_0(e_1)$

Table 1: Measure Bound Rules, including Sekigawa's refinement

bounds on $M_0(e)$ and $M_1(e)$, respectively. These definitions shown in the last two columns of Table 1. Note that we do not have rules for division. Thus, for division free expressions, the function

$$B(e) = \frac{1}{m_0(e)m_1(e)}$$

serves as a root bound function.

The rules shown here is actually a slightly simplified version of his rules.

An Example. Consider the expression

$$e = \sqrt{x} + \sqrt{y} - \sqrt{x + y + 2\sqrt{xy}} \tag{7}$$

where $x = a/b, y = c/d$ and a, b, c, d are L-bit integers. Assume that $\sqrt{xy}$ is computed as $(\sqrt{x})(\sqrt{y})$. We will determine the Measure Bound on $-\lg|e|$ (expressed in terms of L). We call any upper bound for $-\lg|e|$ a **zero bit-bound** for e, because $-\lg|e|$ is the number bits of absolute precision that suffices to determine if $e = 0$.

We fill in the entries of the following table, using our rules for bounding measure. Ignore the last column for $\lg M_0(e)$ for now. It is usually simpler to maintain bounds on $\lg M(e)$ instead of $M(e)$ directly – so that is what we show in the table. The first entry for $\lg M(x)$ is justified in an exercise: $M(a/b) \leq \max\{|a|, |b|\}$ for $a, b \in \mathbb{Z}$.

No.	e	$d(e)$	$\lg M(e)$	$\lg M_0(e)$
1	x, y	1	L	L
2	$\sqrt{x}, \sqrt{y}$	2	L	L
3	$x + y$	1	$2L$	$2L$
4	$\sqrt{x}\sqrt{y}$	4	$4L$	$4L$
5	$\sqrt{x} + \sqrt{y}$	4	$4L + 4$	$4L$
6	$2\sqrt{xy}$	4	$4L + 4$	$4L$
7	$x + y + 2\sqrt{xy}$	4	$12L + 8$	$12L$
8	$\sqrt{x + y + 2\sqrt{xy}}$	8	$12L + 8$	$12L$
9	$\sqrt{x} + \sqrt{y} + \sqrt{x + y + 2\sqrt{xy}}$	8	$80L + 64$	$80L$

Since $|e| \geq 1/M(e)$, we conclude that $-\lg|e| \leq \lg M(e) \leq 80L + 64$. In line 4, the degree $\sqrt{x}\sqrt{y}$ of 4 is obtained from our rules, but it is clearly suboptimal.

Exercises

Exercise 1.1: (i) We have a rule for the measure of rational numbers p/q, and this is clearly tight in case p, q are relatively prime. But show that our measure bound (using the multiplication rule) is sub-optimal for rational input numbers.

(ii) Refine our measure rules for the special case of the division of two algebraic integers, similar to the rule (i). ♢

	e	$U(e)$	$L(e)$
1.	rational a/b	a	b
2.	$e_1 \pm e_2$	$U(e_1)L(e_2) + L(e_1)U(e_2)$	$L(e_1)L(e_2)$
3.	$e_1 \times e_2$	$U(e_1)U(e_2)$	$L(e_1)L(e_2)$
4.	$e_1 \div e_2$	$U(e_1)L(e_2)$	$L(e_1)U(e_2)$
5.	$\sqrt[k]{e_1}$	$\sqrt[k]{U(e_1)}$	$\sqrt[k]{L(e_1)}$

Table 2: BFMS Rules for $U(e)$ and $L(e)$

Exercise 1.2: Determine those algebraic numbers α whose Mahler measure $M(\alpha)$ are not natural numbers. ♢

Exercise 1.3: Determine the Mahler Bound for the expression in Equation (7) where x and y are L-bit rational numbers (i.e., the numerator and denominators are at most L-bit integers). ♢

Exercise 1.4: (i) Determine the Degree-Measure bound for the expression $(a + \sqrt{b})/d$ where a, b, d are (respectively) $3L$-bit, $6L$-bit and $2L$-bit integers.
(ii) Do the same for the difference of two such expressions. ♢

End Exercises

§1.2. The BFMS bound

One of the best constructive zero bounds for the class of radical expressions is from Burnikel et al [5]. We call this the **BFMSS Bound**. However, we begin with the presentation of the simpler version known as the **BFMS Bound** [4]. The bound depends on three parameters, $L(e), L(e), D(e)$ for an expression e. Since $D(e)$ is the usual degree bound, we only show the recursive rules for $L(e), L(e)$ in Table 2.

Conceptually the BFMS approach first transforms a radical expression $e \in Expr(\Omega_2)$ to a quotient of two division-free expressions $U(e)$ and $L(e)$.

If e is division-free, then $L(e) = 1$ and $\texttt{val}(e)$ is an algebraic integer (i.e., a root of some monic integer polynomial). The following lemma is immediate from Table 2:

Lemma 3. $\texttt{val}(e) = \texttt{val}(U(e))/\texttt{val}(L(e))$.

Table 2 should be viewed as transformation rules on expressions. We apply these rules recursive in a bottom-up fashion: suppose all the children v_i (say $i = 1, 2$) of a node v in the expression e has been transformed, and we now have the nodes $U(v_i), L(v_i)$ are available. Then we create the node $U(v), L(v)$ and construct the correspond subexpressions given by the table. The result is still a dag, but not rooted any more. See Figure 1 for an illustration in case the operation at v is $+$.

The transformation $e \Rightarrow (U(e), L(e))$ is only conceptual – we do not really need to compute it. What we do compute are two real parameters $u(e)$ and $l(e)$ are maintained by the recursive rules in Table 3. The entries in this table are "shadows" of the corresponding entries in Table 2. (Where are they different?)

To explain the significance of $u(e)$ and $l(e)$, we define two useful quantities. If α is an algebraic number, define

$$MC(\alpha) := \max_{i=1}^{m} |\alpha_i| \tag{8}$$

where $\alpha_1, \ldots, \alpha_m$ are the conjugates of α. Thus $MC(\alpha)$ is the "maximum conjugate size" of α. We extend this notation in two ways:
(i) If e is an expression, we write $MC(e)$ instead of $MC(\texttt{val}(e))$.

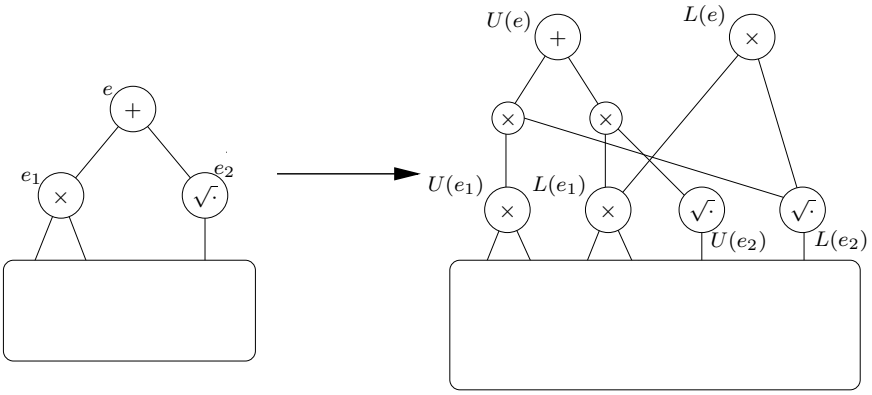

Figure 1: Transforming an expression e to $L(e), U(e)$.

	e	$u(e)$	$l(e)$
1.	rational a/b	$\|a\|$	$\|b\|$
2.	$e_1 \pm e_2$	$u(e_1)l(e_2) + l(e_1)u(e_2)$	$l(e_1)l(e_2)$
3.	$e_1 \times e_2$	$u(e_1)u(e_2)$	$l(e_1)l(e_2)$
4.	$e_1 \div e_2$	$u(e_1)l(e_2)$	$l(e_1)u(e_2)$
5.	$\sqrt[k]{e_1}$	$\sqrt[k]{u(e_1)}$	$\sqrt[k]{l(e_1)}$

Table 3: BFMS Rules for $u(e)$ and $l(e)$

(ii) If $A(X)$ is any polynomial, we write $MC(A(X))$ for the maximum of $|\alpha_i|$ where α_i range over the zeros of $A(X)$. For instance, we have this connection with Mahler measure:

$$M(\alpha) \leq M_0(\alpha) MC(\alpha)^d$$

where $d = \deg(\alpha)$. Using $MC(\alpha)$ and $M_0(\alpha)$, we obtain a basic approach for obtaining zero bounds:

Lemma 4. *If $\alpha \neq 0$ and then*

$$|\alpha| \geq M_0(\alpha)^{-1} MC(\alpha)^{-d+1}$$

where $d = \deg(\alpha)$.

Proof. Let $d = \deg(\alpha)$. If the minimal polynomial of α is $a\prod_{i=1}^{m}(X - \alpha_i)$ then we have $a\prod_i |\alpha_i| \geq 1$. Thus, assuming $\alpha = \alpha_1$,

$$|\alpha| \geq \frac{1}{a\prod_{i=2}^{d} |\alpha_i|} \geq \frac{1}{aMC(\alpha)^{d-1}}.$$

Q.E.D.

The following theorem shows the significance of $u(e), l(e)$.

Theorem 5. *Let $e \in Expr(\Omega_2)$. Then $u(e)$ and $l(e)$ are upper bounds on $MC(U(e))$ and $MC(L(e))$, respectively.*

Proof. The proof amounts to justifying Table 3 for computing $u(e), l(e)$. The base case where e is a rational number is clear. In general, $U(e)$ and $L(e)$ are formed by the rules in Table 2. Consider the case $e = e_1 \pm e_2$ so that

$$U(()e) = U(()e_1)L(()e_2) \pm L(()e_1)U(()e_2).$$

From the theory of resultants, we know that if $\alpha = \beta \circ \gamma$ ($\circ \in \{\pm, \times, \div\}$) then every conjugate of α has the form $\beta' \circ \gamma'$ where β', γ' are conjugates of β, γ (resp.). Thus,

$$MC(U(()e)) = MC(U(()e_1))MC(L(()e_2)) + MC(L(()e_1))MC(U(()e_2)).$$

By induction, $MC(U(()e_i)) \leq u(e_i)$ and $MC(L(()e_i)) \leq l(e_i)$ $(i = 1, 2)$. Hence, $MC(e) \leq u(e_1)l(e_2) + l(e_1)u(e_2) = u(e)$. This justifiies the entry for $u(e)$ on line 2 of Table 3. Similarly, we can justify each of the remaining entries of Table 3. **Q.E.D.**

Finally, we show how the BFMS Rules gives us a zero bound. It is rather similar to Lemma 4, except that we do not need to invoke $M_0(e)$.

Theorem 6. *Let $e \in Expr(\Omega_2)$ and $\mathtt{val}(e) \neq 0$. Then*

$$(u(e)^{D(e)^2-1}l(e))^{-1} \leq |\mathtt{val}(e)| \leq u(e)l(e)^{D(e)^2-1}. \tag{9}$$

If e is division-free,

$$(u(e)^{D(e)-1})^{-1} \leq |\mathtt{val}(e)| \leq u(e). \tag{10}$$

Proof. First consider the division-free case. In this case, $\mathtt{val}(e) = \mathtt{val}(U(e))$. Then $|\mathtt{val}(e)| \leq u(e)$ follows from Theorem 5. The lower bound on $|\mathtt{val}(e)|$ follows from lemma 4, since $M_0(e) = 1$ in the division-free case.

In the general case, we apply the division-free result to $U(e)$ and $L(e)$ separately. However, we need to estimate the degree of $U(e)$ and $L(e)$. We see that in the transformation from e to $U(e), L(e)$, the number of radical nodes in the dag doubles: each $\sqrt[k]{\cdot}$ is duplicated. This means that $\deg(U(e)) \leq \deg(e)^2$ and $\deg(L(e)) \leq \deg(e)^2$. From the division-free case, we conclude that

$$(u(e)^{D(e)^2-1})^{-1} \leq |\mathtt{val}(U(e))| \leq u(e).$$

and

$$(l(e)^{D(e)^2-1})^{-1} \leq |\mathtt{val}(L(e))| \leq l(e).$$

Thus $|\mathtt{val}(e)| = |\mathtt{val}(U(e))/\mathtt{val}(L(e))| \geq (l(e)u(e)^{D(e)^2-1})^{-1}$. The upper bound on $|\mathtt{val}(e)|$ is similarly shown. **Q.E.D.**

Example. Consider the expression $e_k \in Expr(\Omega_2)$ whose value is

$$\alpha_k = \mathtt{val}(e_k) = (2^{2^k} + 1)^{1/2^k} - 2. \tag{11}$$

Note that e_k is not literally the expression shown, since we do not have exponentiation in Ω_2. Instead, the expression begins with the constant 2, squaring k times, plus 1, then taking square-roots k times, and finally minus 2. Thus $u(e_k) = (2^{2^k} + 1)^{1/2^k} + 2 \leq 5$. The degree bound $D(e_k) = 2^k$. Hence the BFMS Bound says

$$|\alpha_k| \geq u(e_k)^{1-2^k} \geq 5^{1-2^k}.$$

How tight is this bound? We have

$$\begin{aligned}
(2^{2^k} + 1)^{1/2^k} - 2 &= 2\left(1 + 2^{-2^k}\right)^{1/2^k} - 2 \\
&= 2 \cdot e^{2^{-k}\ln(1+2^{-2^k})} - 2 \\
&\leq 2 \cdot e^{2^{-k}2^{-2^k}} - 2 \\
&\leq 2\left(1 + 2 \cdot 2^{-k}2^{-2^k}\right) - 2 \\
&= 2^{2-k-2^k}
\end{aligned}$$

	e	$mc(e)$	$m_0(e)$	REMARK
(i)	$a \in C$	$mc(a)$	$m_0(a)$	
(ii)	$e' \pm e''$	$mc(\alpha) + mc(\beta)$	$m_0(e')^{d''} m_0(e'')^{d'}$	$\deg(e') \leq d'$
(iii)	$e' \times e''$	$mc(\alpha)mc(\beta)$	$m_0(e')^{d''} m_0(e'')^{d'}$	$\deg(e'') \leq d''$
(iv)	$\sqrt[k]{e'}$	$\sqrt[k]{mc(e')}$	$m_0(e')$	

Table 4: Measure-BFMS Rules using $mc(e)$ and $m_0(e)$

using $\ln(1+x) \leq x$ if $x > -1$ and $e^2 \leq 1 + 2x$ if $0 \leq x \leq 1/2$. We also have

$$\begin{aligned}(2^{2^k}+1)^{1/2^k} - 2 &= 2 \cdot e^{2^{-k}\ln(1+2^{-2^k})} - 2 \\ &\geq 2 \cdot e^{2^{-k}2^{-2^k-1}} - 2 \\ &\geq 2\left(1 + 2^{-k}2^{-2^k-1}\right) - 2 \\ &\geq 2^{-k-2^k}\end{aligned}$$

using $e^x \geq 1 + x$. Hence $\alpha_k = \Theta(2^{-k-2^k})$. This example shows that the BFMS bound is, in a certain sense, asymptotically tight for the class of division-free expressions over Ω_2.

§1.3. Improvements on the BFMS bound

The root bit-bound in (9) is quadratic in $D(e)$, while in (10) it is linear in $D(e)$. This quadratic factor can become a serious efficiency issue. Consider a simple example: $e = (\sqrt{x} + \sqrt{y}) - \sqrt{x + y + 2\sqrt{xy}}$ where x, y are L-bit integers. Of course, this expression is identically 0 for any x, y. The BFMS bound yields a root bit-bound of $7.5L + \mathcal{O}(1)$ bits. But in case, x and y are viewed as rational numbers (with denominator 1), the bit-bound becomes $127.5L + \mathcal{O}(1)$. This example shows that introducing rational numbers at the leaves of expressions has a major impact on the BFMS bound. In this section, we introduce two techniques to overcome division.

The Measure-BFMS Bound. The first technique applies division-free expressions, but where the input numbers need not be algebraic integers (we can think of this as allowing division at the leaves). For instance, the input numbers can be rational numbers.

The basic idea is to exploit Lemma 4. Hence we would like to maintain upper bounds on $MC(\alpha)$ and $M_0(\alpha)$. Let

$$\Omega = \{\pm, \times, \sqrt[k]{\cdot}\} \cup C \tag{12}$$

where C is some set of algebraic numbers. Suppose e is an expression over Ω; so e is division-free. However C may contain rational numbers that implicitly introduce division. We define the numerical parameters $mc(e)$ and $m_0(e)$ according to the recursive rules in Table 4.

Table 4, gives the recursive rules for computing $mc(e), m_0(e)$. Note that as the base case, we assume the ability to compute upper bounds on $MC(a)$ and $M_0(a)$ for $a \in C$.

Lemma 7. *For any expression e over $\{\pm, \times, \sqrt[k]{\cdot}\} \cup C$, we have $MC(e) \leq mc(e)$ and $M_0(e) \leq m_0(e)$ where $mc(e)$ and $m_0(e)$ are given by Table 4.*

Proof. We justify each line of Table 4. Line (i) is immediate by definition.

(ii) Let the minimal polynomial for α, β be $a\prod_{i=1}^{m}(X - \alpha_i)$ and $b\prod_{j=1}^{n}(X - \beta_j)$, respectively. Then the minimal polynomial of $\alpha \pm \beta$ divides $R(X) = a^n b^m \prod_i \prod_j (X - \alpha_i \mp \beta_j)$. Thus each conjugate ξ of $\alpha \pm \beta$ has the form $\alpha_i \pm \beta_j$ for some i, j. Thus $|\xi| \leq |\alpha_i| + |\beta_j|$. This proves $MC(\alpha \pm \beta) \leq mc(\alpha) + mc(\beta)$. The

inequality $M_0(\alpha \pm \beta) \leq a^n b^m \leq m_0(\alpha)^n m_0(\beta)^m$ follows from the fact that the leading coefficient of the minimal polynomial divides the leading coefficient of $R(X)$.

The same proof applies for $e = e_1 e_2$. For (iii), if $A(X)$ is the minimal polynomial for α then the minimal polynomial for $\sqrt[k]{\alpha}$ divides $A(X^k)$, so $MC(\alpha) \leq MC(A(X^k))$. However, $MC(A(X^k)) = \sqrt[k]{MC(A(X))} = \sqrt[k]{MC(\alpha)}$. Finally, we have $M_0(\sqrt[k]{\alpha}) \leq a \leq M_0(\alpha)$. **Q.E.D.**

By Lemma 4, we conclude that

$$e \neq 0 \Rightarrow |e| \geq \frac{1}{M_0(e) mc(e)^{D(e)-1}}. \tag{13}$$

Such a bound has features of Measure Bound as well as of the BFMS Bound; so we call it the "Measure-BFMS Bound".

The BFMSS Bound. Returning to the case of radical expressions, we introduce another way to improve on BFMS. To avoid the doubling of radical nodes in the $e \mapsto (U(e), L(e))$ transformation, we change the rule in the last row of Table 3 as follows. When $e = \sqrt[k]{e_1}$, we use the alternative rule

$$u(e) = \sqrt[k]{u(e_1) l(e_1)^{k-1}}, \quad l(e) = l(e_1). \tag{14}$$

But one could equally use

$$u(e) = u(e_1), \quad l(e) = \sqrt[k]{u(e_1)^{k-1} l(e_1)}.$$

Yap noted that by using the symmetrized rule

$$u(e) = \min\{\sqrt[k]{u(e_1) l(e_1)^{k-1}}, u(e_1)\}, \qquad l(e) = \min\{l(e_1), \sqrt[k]{u(e_1)^{k-1} l(e_1)}\},$$

the new bound is provably never worse than the BFMS bound. The BFMSS Bound also extends the rules to support general algebraic expressions (Ω_4 expressions). NOTE: in the absence of division, the BFMSS and BFMS rules coincide.

Comparison. We consider expressions over a division-free set Ω of operators, as in (12). Two important examples are:

- Expressions can have rational input numbers (in particular binary floats or decimal numbers). See [Pion-Yap].
- Expressions where the leaves could have RootOf(P, i) operators. This is the case with Core Library Version 1.6.

It follows that Theorem 5 is still true. However, what is the replacement for Theorem 6? Using Lemma 4 and Lemma 7 we can obtain more effective bounds.

Let us see how BFMS, BFMSS and Measure-BFMS Bounds perform for the expression (7). In Table 3, we show the parameters $u(e), l(e)$ as defined for the BFMS Bound. In the next two columns, we show their variants (here denoted $uu(e)$ and $ll(e)$) as defined for the BFMSS Bound.

The BFMS Bound gives

$$-\lg|e| \leq (d(e)^2 - 1)\lg u(e) + \lg l(e) = 63(5L+4)/2 + 5L/2 < 160L + 126.$$

But the BFMSS Bound gives

$$-\lg|e| \leq (d(e) - 1)\lg uu(e) + \lg ll(e) \leq 7(4L+2) + 4L = 32L + 14.$$

No.	e	$d(e)$	$\lg u(e)$	$\lg l(e)$	$\lg uu(e)$	$\lg ll(e)$	$mc(e)$
1	x, y	1	L	L	L	L	L
2	$\sqrt{x}, \sqrt{y}$	2	$L/2$	$L/2$	L	L	$L/2$
3	$x + y$	1	$2L + 1$	$2L$	$2L + 1$	$2L$	$L + 1$
4	$\sqrt{x}\sqrt{y}$	4	L	L	$2L$	$2L$	L
5	$\sqrt{x} + \sqrt{y}$	4	$L + 1$	L	$2L + 1$	$2L$	$(L + 2)/2$
6	$2\sqrt{xy}$	4	$L + 1$	L	$2L + 1$	$2L$	$L + 1$
7	$x + y + 2\sqrt{xy}$	4	$3L + 2$	$3L$	$4L + 2$	$4L$	$L + 2$
8	$\sqrt{x + y + 2\sqrt{xy}}$	8	$(3L + 2)/2$	$3L/2$	$2L + 1$	$2L$	$(L + 2)/2$
9	$\sqrt{x} + \sqrt{y} - \sqrt{x + y + 2\sqrt{xy}}$	8	$(5L + 4)/2$	$5L/2$	$4L + 2$	$4L$	$(L + 4)/2$

Table 5: BFMS and BFMSS on example

To apply the Measure-BFMS rule, we could use the fact that

$$MC(e) \leq u(e) \leq (5L + 2)/2$$

(by first column of Table

$$M_0(e) \leq 80L$$

(by last column of table in (i)). Hence

$$-\lg |e| \leq 7(5L + 2)/2 + 80L = 97.5L + 7.$$

But we can directly compute an upper bound on $MC(e)$ using the rules in Table 3. This is shown in the last column of Table . This gives $MC(e) \leq mc(e) \leq (L + 4)/2$. Then

$$-\lg |e| \leq 7(L + 4)/2 + 80L = 83.5L + 3.5.$$

In the next section, we consider the Conjugate Bound which is an extension of the Measure-BFMS approach: for this example, it yields $-\lg |e| \leq 28L + 60$.

Exercises

Exercise 1.5: Show an expression e involving L-bit integers where the application of Theorem 5 is asymptotically better than that of measure or BFMSS bounds. ♢

Exercise 1.6: Prove that the BFMSS Bound is never smaller than BFMS Bound. ♢

End Exercises

§1.4. Eigenvalue Bound

This bound adopts an interesting approach based on matrix eigenvalues [12]. Let $\Lambda(n, b)$ denote the set of eigenvalues of $n \times n$ matrices with integer entries with absolute value at most b. It is easy to see that $\Lambda(n, b)$ is a finite set of algebraic integers. Moreover, if $\alpha \in \Lambda(n, b)$ is non-zero then $|\alpha| \geq (nb)^{1-n}$. Scheinerman gives a constructive zero bound for division-free radical expressions e by maintaining two parameters, $n(e)$ and $b(e)$, satisfying the property that the value of e is in $\Lambda(n(e), b(e))$. These recursive rules are given by Table 6.

Note that the rule for $\sqrt{cd}$ is rather special, but it can be extremely useful. In Rule 6, the polynomial $\overline{P}(x)$ is given by $\sum_{i=0}^{d} |a_i| x^i$ when $P(x) = \sum_{i=0}^{d} a_i x^i$. This rule is not explicitly stated in [12], but can be deduced from an example he gave. An example given in [12] is to test whether $\alpha = \sqrt{2} + \sqrt{5 - 2\sqrt{6}} - \sqrt{3}$ is zero. Scheinerman's bound requires calculating α to 39 digits while the BFMS bound says 12 digits are enough.

	e	$n(e)$	$b(e)$
1.	integer a	1	$\|a\|$
2.	$\sqrt{cd}$	2	$\max\{\|c\|, \|d\|\}$
3.	$e_1 \pm e_2$	$n_1 n_2$	$b_1 + b_2$
4.	$e_1 \times e_2$	$n_1 n_2$	$b_1 b_2$
5.	$\sqrt[k]{e_1}$	kn_1	b_1
6.	$P(e_1)$	n_1	$\overline{P}(n_1 b_1)$

Table 6: Eigenvalue Rules

§1.5. Conjugate Bound

The "conjugate bound" [8] is an extension of the Measure-BFMS bound above expressions with division. This approach can give significantly better performance than BFMS in many expressions involving divisions and root extractions. Because of division, we also maintain upper bounds $\mathrm{tc}(e)$, $M(e)$ on the tail $\mathtt{tail}(e)$ and $M(e)$. Here the tail coefficient $\mathtt{tail}(e)$ is defined as the constant term of the irreducible polynomial $\mathrm{Irr}(e)$.

Table 7 gives the recursive rules to maintain $M_0(e), \mathrm{tc}(e)$ and $M(e)$.

	e	$\mathrm{lc}(e)$	$\mathrm{tc}(e)$	$M(e)$
1.	rational $\frac{a}{b}$	$\|b\|$	$\|a\|$	$\max\{\|a\|, \|b\|\}$
2.	$\mathrm{Root}(P)$	$\|\mathtt{lead}P\|$	$\|\mathtt{tail}P\|$	$\|P\|_2$
3.	$e_1 \pm e_2$	$\mathrm{lc}_1^{D_2}\mathrm{lc}_2^{D_1}$	$M_1^{D_2} M_2^{D_1} 2^{D(e)}$	$M_1^{D_2} M_2^{D_1} 2^{D(e)}$
4.	$e_1 \times e_2$	$\mathrm{lc}_1^{D_2}\mathrm{lc}_2^{D_1}$	$\mathrm{tc}_1^{D_2}\mathrm{tc}_2^{D_1}$	$M_1^{D_2} M_2^{D_1}$
5.	$e_1 \div e_2$	$\mathrm{lc}_1^{D_2}\mathrm{tc}_2^{D_1}$	$\mathrm{tc}_1^{D_2}\mathrm{lc}_2^{D_1}$	$M_1^{D_2} M_2^{D_1}$
6.	$\sqrt[k]{e_1}$	lc_1	tc_1	M_1
7.	e_1^k	lc_1^k	tc_1^k	M_1^k

Table 7: Recursive rules for $\mathrm{lc}(e)$ (and associated $\mathrm{tc}(e)$ and $M(e)$)

The upper bounds on conjugates, $MC(e)$, are obtained through resultant calculus and standard interval arithmetic techniques. It turns out that it is necessary to maintain a lower bound $\underline{\nu}(e)$ on the conjugates at the same time. The recursive rules to maintain these two bounds are given in Table 8.

	e	MC(e)	$\underline{\nu}(e)$
1.	rational $\frac{a}{b}$	$\|\frac{a}{b}\|$	$\|\frac{a}{b}\|$
2.	$\mathrm{Root}(P)$	$1 + \|P\|_\infty$	$(1 + \|P\|_\infty)^{-1}$
3.	$e_1 \pm e_2$	$MC(e_1) + MC(e_2)$	$\max\{M(e)^{-1}, (MC(e)^{D(e)-1}\mathrm{lc}(e))^{-1}\}$
4.	$e_1 \times e_2$	$MC(e_1)MC(e_2)$	$\underline{\nu}(e_1)\underline{\nu}(e_2)$
5.	$e_1 \div e_2$	$MC(e_1)/\underline{\nu}(e_2)$	$\underline{\nu}(e_1)/MC(e_2)$
6.	$\sqrt[k]{e_1}$	$\sqrt[k]{MC(e_1)}$	$\sqrt[k]{\underline{\nu}(e_1)}$
7.	e_1^k	$MC(e_1)^k$	$\underline{\nu}(e_1)^k$

Table 8: Recursive rules for bounds on conjugates

Finally, we obtain the new zero bound as follows: Given an Ω_3-expression e, if $\mathtt{val}(e) \downarrow$ and $e \neq 0$, then we obtain the lower bound from Lemma 4. This bound was implemented the `Core Library` and experiments show that it can achieve significant speedup over previous bounds in the presence of division [8].

§1.6. The Factoring Method for Root Bounds

Pion and Yap [11] introduced a root-bound technique based on the following idea: maintain zero bounds for an expression e in the factored form, $b = b_1 b_2$ where b_i $(i = 1, 2)$ is a zero bound for e_i and $e = e_1 e_2$. If b_i is obtained using method X (X being one of the above methods), and the factorization is carefully chosen, then $b_1 b_2$ could be a better bound than what X would have produced for e directly. The catch is the method has no advantage unless such a factorization for e exists and can be easily found. Fortunately, there is a class of predicates for which this approach wins: division-free predicates (e.g., determinants) in which the input numbers are **k-ary rationals**, i.e., numbers of the form nk^m where $n, m \in \mathbb{Z}$. This is an important class as the majority of real world input numbers are k-ary rationals for $k = 2$ or $k = 10$. For such expressions, we maintain bounds for the factorization $e = e_1 e_2$ where e_1 is division-free (but may have square roots) and $e_2 = k^v$ for some $v \in \mathbb{Z}$. Our technique is orthogonal to the various zero bounds which we already discussed because each of the root bounds can, in principle, use this factorization technique. This has been implemented in the `Core Library` for the BFMSS Bound and the Measure Bound, resulting in significant improvement for zero bounds in, for instance, determinants with k-ary rational inputs.

§1.7. Comparison of Bounds

Comparisons between various constructive zero bounds can be found in [4, 8]. In general, a direct comparison of the above zero bounds is a difficult task because of the different choice of parameters and bounding functions used. Following the tact in [8], we compare their performance on various special subclasses of algebraic expressions. We should note that there are currently three zero bounds, the BFMSS, Li-Yap and Measure Bounds, that are not dominated by any other methods. In particular, these three are mutually incomparable.

1. For division-free radical expressions, the BFMS bound is never worse than all the other bounds. Moreover, for this special class of expressions, Li-Yap bound is identical to the BFMS bound.

2. For general algebraic expressions, in terms of root bit-bound, Li-Yap bound is at most $D \cdot M$ where D is the degree bound, and M is the root bit-bound from the degree-measure bound.

3. Considering the sum of square roots of rational numbers (a common problem in solving the shortest Euclidean path problem), it can be shown that each of Li-Yap bound and the degree-measure bound can be better than the other depending on different parameters about the expressions. But both of them are always better than the BFMS bound.

4. Given a radical expression e with rational values at the leaves, if e has no divisions and shared radical nodes, Li-Yap bound for e is never worse than the BFMS bound, and can be better in many cases.

5. A critical test in Fortune's sweepline algorithm is to determine the sign of the expression $e = \frac{a+\sqrt{b}}{d} - \frac{a'+\sqrt{b'}}{d'}$ where a's, b's and d's are $3L$-, $6L$- and $2L$-bit integers, respectively. The BFMS bound requires $(79L + 30)$ bits and the degree-measure (D-M) bound needs $(64L + 12)$ bits. Li-Yap root bit-bound improves them to $(19L+9)$ bits. We generate some random inputs with different L values which always make $e = 0$, and put the timings (in seconds) of the tests in Table 9. The experiments are performed on a Sun UltraSPARC with a 440 MHz CPU and 512MB main memory.

L	10	20	50	100	200
NEW	0.01	0.03	0.12	0.69	3.90
BFMS	0.03	0.24	1.63	11.69	79.43
D-M	0.03	0.22	1.62	10.99	84.54

Table 9: Timings for Fortune's expression

§1.8. Treatment of Special Cases

Root separation bounds. In both the `Core Library` and `LEDA`, the comparison of two expressions α and β is obtained by computing the zero bound of $\alpha - \beta$. However, more efficient techniques can be used. If $P(X) \in \mathbb{C}[X]$ is a non-zero polynomial polynomial, sep(P) denotes the minimum $|\alpha_i - \alpha_j|$ where $\alpha_i \neq \alpha_j$ range over all pairs of complex roots of P. When P has less than two distinct roots, define $\text{sep(P)} = \infty$. Suppose $A(X)$ and $B(X)$ are the minimal polynomials for α and β, then $|\alpha - \beta| \geq \text{sep(AB)}$. If we maintain upper bounds d, d' on the degrees of A and B, and upper bounds h, h' on the heights of A and B, a root separation bound for $A(X)B(X)$ (which need not be square-free) is given by

$$|\alpha - \beta| \geq \left[2^{(n+1)/2}(n+1)hh'\right]^{-2n} \tag{15}$$

where $n = d + d'$ (see Corollary 6.33 in [15, p. 176,173] and use the fact that $\|AB\|_2 \leq (n+1)hh'$). The advantage of using (15) is that the root bit bound here is linear in $d + d'$, and not dd', as would be the case if we use resultant calculus. We compute α and β to an absolute error $< \text{sep(AB)}/4$ each, then declare them to be equal iff their approximations differ by $\leq \text{sep(AB)}/2$ from each other. Otherwise, the approximations tell us which number is larger. Note that this approach not fit into our recursive zero bound framework (in particular, it does not generate bounds for a new minimal polynomial).

Zero test. Zero testing is the special case of sign determination in which we want to know whether an expression is zero or not. Many predicates in computational geometry programs are really zero tests (e.g. detection of degeneracy, checking if a point lies on a hyperplane). In other applications, even though we need general sign determination, the zero outcome is very common. For instance, in the application of EGC to theorem proving [14], true conjectures are equivalent to the zero outcome. In our numerical approach based on zero bounds, the complexity of sign determination is determined by the zero bound when the outcome is zero. Since zero bounds can be overly pessimistic, such tests can be extremely slow. Hence it is desirable to have an independent method of testing if an expression is zero. Such a zero test can be used as a filter for the sign determination algorithm. Only when the filter detects a non-zero do we call the iterative precision numerical method.

Yap and Blömer [3] observed that for expressions of the form $e = \sum_{i=1}^{n} a_i\sqrt{b_i}$ ($a_i \in \mathbb{Z}, b_i \in \mathbb{N}$), zero testing is deterministic polynomial time, while the sign determination problem is not known to be polynomial time. Blömer [3, 1] extended this to the case of general radicals using a theorem of Siegel; he also [2] gave a probabilistic algorithm for zero test. When the radicals are nested, we can apply denesting algorithms [6, 7]. Note that these methods are non-numerical.

Exercises

Exercise 1.7: Let $a > 0$. Show that $a\max\{1, 1/a\} = \max\{1, a\}$ and $(1/a)\max\{1, a\} = \max\{1, 1/a\}$ ♢

Exercise 1.8: Show the conjugates of α are distinct. ♢

Exercise 1.9: Show that $M(A) = \exp\left[\int_0^1 \log|A(e(\theta))|d\theta|\right]$. ♢

End Exercises

References

[1] J. Blömer. Computing sums of radicals in polynomial time. *IEEE Foundations of Computer Sci.*, 32:670–677, 1991.

[2] J. Blömer. A probabilistic zero-test for expressions involving roots of rational numbers. *Proc. of the Sixth Annual European Symposium on Algorithms*, pages 151–162, 1998. LNCS 1461.

[3] J. Blömer. *Simplifying Expressions Involving Radicals*. PhD thesis, Free University Berlin, Department of Mathematics, October, 1992.

[4] 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.

[5] C. Burnikel, S. Funke, K. Mehlhorn, S. Schirra, and S. Schmitt. A separation bound for real algebraic expressions. In *9th ESA*, volume 2161 of *Lecture Notes in Computer Science*, pages 254–265. Springer, 2001. To appear, Algorithmica.

[6] G. Horng and M. D. Huang. Simplifying nested radicals and solving polynomials by radicals in minimum depth. *Proc. 31st Symp. on Foundations of Computer Science*, pages 847–854, 1990.

[7] S. Landau. Simplification of nested radicals. *SIAM Journal of Computing*, 21(1):85–110, 1992.

[8] C. Li and C. Yap. A new constructive root bound for algebraic expressions. In *12th SODA*, pages 496–505, Jan. 2001.

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

[10] M. Mignotte and D. Ştefănescu. *Polynomials: An Algorithmic Approach*. Springer, Singapore, 1999.

[11] S. Pion and C. Yap. Constructive root bound method for k-ary rational input numbers. In *19th SCG*, pages 256–263, San Diego, California., 2003. Accepted, Theoretical Computer Science (2006).

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

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

[14] D. Tulone, C. Yap, and C. Li. Randomized zero testing of radical expressions and elementary geometry theorem proving. In J. Richter-Gebert and D. Wang, editors, *Proc. 3rd Int'l. Workshop on Automated Deduction in Geometry (ADG 2000)*, volume 2061 of *Lecture Notes in Artificial Intelligence*, pages 58–82. Springer, 2001. Zurich, Switzerland.

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

[16] C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, pages 452–492. World Scientific Press, Singapore, 2nd edition, 1995.