Algorithmica (2000) 27: 87–99
DOI: 10.1007/s004530010005

Algorithmica
© 2000 Springer-Verlag New York Inc.

A Strong and Easily Computable Separation Bound for Arithmetic Expressions Involving Radicals

C. Burnikel,[1] R. Fleischer,[1] K. Mehlhorn,[1] and S. Schirra[1]

Abstract. We consider arithmetic expressions over operators $+$, $-$, $*$, $/$, and $\sqrt[k]{\ }$, with integer operands. For an expression E having value ξ, a separation bound $sep(E)$ is a positive real number with the property that $\xi \neq 0$ implies $|\xi| \geq sep(E)$. We propose a new separation bound that is easy to compute and stronger than previous bounds.

Key Words. Exact geometric computation, Separation bound.

1. Introduction. The evaluation of a test in a computer program frequently amounts to determining the sign of an arithmetic expression, e.g., if p, q, and r are points in the plane, then

$$\sqrt{(p_x - r_x)^2 + (p_y - r_y)^2} \leq \sqrt{(q_x - r_x)^2 + (q_y - r_y)^2}$$

determines whether p is at least as close to r as q. Section 4.2 discusses a more substantial example.

We consider expressions with operators

$$+, -, *, /, \text{ and } \sqrt[k]{\ },$$

and integer operands. This class of expressions is particularly relevant for geometric computations [2], [3], [9], [15], [17]. An expression E is given as a directed acyclic graph (dag) whose source nodes are labeled by integers and whose internal nodes are labeled by operators from $+$, $-$, $*$, $/$, and $\sqrt[k]{\ }$. An expression is called *division-free* if it contains no nodes labeled with $/$.

For expression E we construct a quotient $E = U(E)/L(E)$ of division-free expressions $U(E)$ and $L(E)$. In Table 1 we define $U(E)$ and $L(E)$ inductively on the structure of E. Furthermore, we define nonnegative real numbers $u(E)$ and $l(E)$ in Table 2. The numbers $u(E)$ and $l(E)$ are bounds on the absolute values of $U(E)$ and $L(E)$, respectively, obtained by replacing every subtraction in E by an addition. Note that $u(U(E)) = u(E)$ and $u(L(E)) = l(E)$, and $l(E) = 1$ for division-free expressions.

While there is no doubt about the meaning of a rule like $U(E) = U(E_1) \cdot U(E_2)$ for $E = E_1 \cdot E_2$ if E is given as an expression tree, it is not immediately clear what such a rule means if E is given as an expression dag involving common subexpressions. The

[1] Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. {burnikel, rudolf, mehlhorn, stschirr}@mpi-sb.mpg.de.

Received February 24, 1997; revised August 3, 1998. Communicated by S. Fortune.

Table 1. Inductive rules for $U(E)$ and $L(E)$.

E	$U(E)$	$L(E)$
Integer n	n	1
$E_1 \pm E_2$	$U(E_1) \cdot L(E_2) \pm L(E_1) \cdot U(E_2)$	$L(E_1) \cdot L(E_2)$
$E_1 \cdot E_2$	$U(E_1) \cdot U(E_2)$	$L(E_1) \cdot L(E_2)$
E_1 / E_2	$U(E_1) \cdot L(E_2)$	$L(E_1) \cdot U(E_2)$
$\sqrt[k]{E_1}$	$\sqrt[k]{U(E_1)}$	$\sqrt[k]{L(E_1)}$

rules on dags should be interpreted as follows. Compute a topological order on the nodes of the dag and apply the rules to the nodes in this order. Then the result for the at most two subexpressions linked to a node is already computed, when the value of the node is computed. Combine these results as defined by the rule corresponding to the current node operation. Note the difference in the recursive rules for $u()$, $l()$ and $U()$, $L()$. The former rules operate on real numbers, the latter rules operate on dags. For the rules for $U()$ and $L()$ it is important not to process common subexpressions more than once to avoid an uncontrolled increase in the number of $\sqrt[k]{\;}$-nodes in the dags.

We denote the value of an expression E by $\text{val}(E)$. The number $\xi = \text{val}(E)$ is algebraic, i.e., there is a polynomial $p \in \mathbb{Z}(X)$ such that $p(\xi) = 0$. Among the polynomials in $\mathbb{Z}(X)$ having root ξ there exists a unique irreducible polynomial m_ξ, called the minimal polynomial of ξ. The degree $deg(\xi)$ of ξ is defined as the degree of the polynomial m_ξ. If E involves r radical nodes of indices $k_1, \ldots, k_r$, respectively, then we define $D(E) = \prod_{i=1}^{r} k_i$. Note that $deg(\xi)$ is bounded from above by $D(E)$.

First, we state our main theorem in an abstract form involving $deg(\xi)$ which is not easily computable.

THEOREM 1. *If E is a division-free expression whose value ξ is nonzero, then*

$$\left(u(E)^{deg(\xi)-1}\right)^{-1} \le |\xi| \le u(E).$$

The following corollaries involve $D(E)$ in place of $deg(\text{val}(E))$. These are the versions on which the actual automatic computation of separation bounds is based.

COROLLARY 2. *Let E be a division-free expression whose value ξ is nonzero. Then*

$$\left(u(E)^{D(E)-1}\right)^{-1} \le |\xi| \le u(E).$$

Table 2. Inductive rules for $u(E)$ and $l(E)$.

E	$u(E)$	$l(E)$
Integer n	$\|n\|$	1
$E_1 \pm E_2$	$u(E_1) \cdot l(E_2) + l(E_1) \cdot u(E_2)$	$l(E_1) \cdot l(E_2)$
$E_1 \cdot E_2$	$u(E_1) \cdot u(E_2)$	$l(E_1) \cdot l(E_2)$
E_1 / E_2	$u(E_1) \cdot l(E_2)$	$l(E_1) \cdot u(E_2)$
$\sqrt[k]{E_1}$	$\sqrt[k]{u(E_1)}$	$\sqrt[k]{l(E_1)}$

For expressions involving divisions we get

COROLLARY 3. *Let E be an expression whose value ξ is nonzero. Then*

$$\left(l(E)u(E)^{D^2(E)-1}\right)^{-1} \leq |\xi| \leq u(E)l(E)^{D^2(E)-1}.$$

For the special case in which all radicals are square root operations we have in particular: If E is an expression involving r nodes with square root operation and the value ξ of E is nonzero, then

$$\left(l(E)u(E)^{2^{2r}-1}\right)^{-1} \leq |\xi| \leq u(E)l(E)^{2^{2r}-1}.$$

If E is division-free and $\xi = \mathrm{val}(E) \neq 0$, then

$$\left(u(E)^{2^r-1}\right)^{-1} \leq |\xi| \leq u(E).$$

There are examples where these bounds are nearly optimal. Consider

$$E = \left(2^{2^k}+1\right)^{1/2^k} - 2,$$

i.e., the number 2 is squared k times, 1 is added, square roots are taken k times, and finally 2 is subtracted. $\xi = \mathrm{val}(E)$ is a number of algebraic degree 2^k and $u(E) = (2^{2^k}+1)^{1/2^k} + 2 \leq 5$, hence $|\xi| \geq 5^{1-2^k}$ by Theorem 1. On the other hand, $|\xi| \leq 2^{-2^k}$ for $k \geq 2$ since

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

Here we used the inequalities $\ln(1+x) \leq x$ if $x > -1$ and $e^x \leq 1 + 2x$ if $0 \leq x \leq \frac{1}{2}$.

There is a significant amount of literature on separation bounds in computer algebra [5], [8], [13], see also [16], as well as in computational geometry [2], [3], [9], [15]. Some of the work in computer algebra has concentrated on root separation bounds for systems of polynomial equations of which our problem is a special case. We show in Section 3 that the bound of Theorem 1 is never worse than the general bounds and sometimes significantly better. The work in computational geometry has concentrated on providing root separation bounds for specific geometric predicates. For example, [3] discusses the predicate which decides for a circle passing through a given point and touching two given lines whether it intersects, touches, or misses a third given line and shows a separation bound of $2^{-(32b+O(1))}$ if all endpoints of the line segments involved have integer coordinates bounded by 2^b. In [2] this bound is improved to $2^{-(24b+O(1))}$. In [9] it is shown that a certain part of the reasoning in the two aforementioned papers can be specified as rewrite rules. In this paper we go one step further and show that an even larger part of the reasoning can be fully automatized. We have to admit, however, that some of our bounds are slightly weaker than those derived in [2] and [3] for the specific expressions involved in incircle tests, see Section 4.

We give a proof of Theorem 1 in Section 2, compare it with previous work in Section 3, and discuss its applications to computational geometry and the number type `real` of LEDA [4], [11] in Section 4.

2. The Proof.

We give the proof of our main theorem. The upper bound is obvious. We turn to the lower bound. Let E be a division-free expression. Our first step is a constructive proof that $\xi = \mathrm{val}(E)$ is an algebraic integer, which can be found e.g., in [7] and [10]. For the sake of completeness we sketch the proof.

THEOREM 4. *Let* $p_A(X) = \prod_{i=1}^{n}(X-\alpha_i) = \sum_{i=0}^{n-1} a_i X^i + X^n$ *and* $p_B(X) = \prod_{j=1}^{m}(X-\beta_j) = \sum_{j=0}^{m-1} b_j X^j + X^m$ *be polynomials with integral coefficients. Then for every operation* op $\in \{+, -, *\}$ *the polynomial*

$$p_{A\,\mathrm{op}\,B}(X) = \prod_{i=1}^{n}\prod_{j=1}^{m}(X - (\alpha_i \ \mathrm{op}\ \beta_j))$$

has integral coefficients as well.

PROOF. We use the well-known theorem on the elementary symmetric functions.

THEOREM 5. *Let R be a commutative ring with unity and let* $\sigma_1^{(t)} = t_1 + t_2 + \cdots + t_n$, $\sigma_2^{(t)} = t_1t_2 + t_1t_3 + \cdots + t_{n-1}t_n, \ldots, \sigma_n^{(t)} = t_1t_2\cdots t_n$ *be the elementary symmetric functions in* $t_1, \ldots, t_n$, *i.e.,*

$$\prod_{j=1}^{n}(X - t_i) = \sum_{j=0}^{n-1} \sigma_{n-j}^{(t)} X^j + X^n.$$

Let p be a polynomial in $t_1, \ldots, t_n$ *with coefficients in R which is symmetric in* $t_1, \ldots, t_n$, *i.e., for any permutation π of* $[1..n]$ *we have*

$$p(t_1, \ldots, t_n) = p(t_{\pi(1)}, \ldots, t_{\pi(n)}).$$

Then there is a polynomial q with coefficients in R such that

$$p(t_1, \ldots, t_n) = q(\sigma_1^{(t)}, \ldots, \sigma_n^{(t)}).$$

A proof can be found, e.g., in [14]. Now let op $\in \{+, -, *\}$. We apply the theorem on symmetric functions to the polynomial

$$f(X, \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m) = \prod_{i=1}^{n}\prod_{j=1}^{m}(X - (\alpha_i \ \mathrm{op}\ \beta_j))$$

which is symmetric in $\alpha_1, \ldots, \alpha_n$ as well as in $\beta_1, \ldots, \beta_m$. Since the polynomial $f(X, \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m)$ is symmetric in $\beta_1, \ldots, \beta_m$, there is a polynomial $g(X, \alpha_1, \ldots, \alpha_n, \sigma_1^{(\beta)}, \ldots, \sigma_m^{(\beta)})$ such that

$$g(X, \alpha_1, \ldots, \alpha_n, \sigma_1^{(\beta)}, \ldots, \sigma_m^{(\beta)}) = \prod_{i=1}^{n} \prod_{j=1}^{m} (X - (\alpha_i \text{ op } \beta_j)).$$

Since $g(X, \alpha_1, \ldots, \alpha_n, \sigma_1^{(\beta)}, \ldots, \sigma_m^{(\beta)})$ is symmetric in $\alpha_1, \ldots, \alpha_n$, we can apply the theorem on the elementary symmetric functions once more to get a polynomial $h(X, \sigma_1^{(\alpha)}, \ldots, \sigma_n^{(\alpha)}, \sigma_1^{(\beta)}, \ldots, \sigma_m^{(\beta)})$ such that

$$h(X, \sigma_1^{(\alpha)}, \ldots, \sigma_n^{(\alpha)}, \sigma_1^{(\beta)}, \ldots, \sigma_m^{(\beta)}) = \prod_{i=1}^{n} \prod_{j=1}^{m} (X - (\alpha_i \text{ op } \beta_j)).$$

We have $\sigma_1^{(\alpha)} = a_{n-1}, \ldots, \sigma_n^{(\alpha)} = a_0$ and $\sigma_1^{(\beta)} = b_{m-1}, \ldots, \sigma_m^{(\beta)} = b_0$. By assumption, they all are integral, and hence h is a polynomial in X with integral coefficients. □

The lemma can be applied to construct explicitly a polynomial $p_E \in \mathbb{Z}(X)$ with root ξ.

LEMMA 6. *There exists a polynomial*

$$p_E(X) = \prod_i (X - e_i) \in \mathbb{Z}(X)$$

such that $p_E(\xi) = 0$ *and* $|e_i| \leq u(E)$ *for all roots* e_i *of* $p_E(X)$.

PROOF. We use induction on the structure of E. If E is an integer constant a, then the polynomial $p_E(X) = X - a$ satisfies the conditions of the lemma. If $E = A \text{ op } B$ for op $\in \{+, -, *\}$, then we know by induction that there exist polynomials $p_A(X) = \prod_{i=1}^{n}(X - \alpha_i)$ and $p_B(X) = \prod_{j=1}^{m}(X - \beta_j)$ with roots val(A) and val(B), respectively. Then we choose $p_E(X) = p_{A \text{ op } B}(X)$. By Theorem 4, $p_{A \text{ op } B}(X)$ has integral coefficients. Furthermore, $p_{A \text{ op } B}(X)$ has root val$(E) =$ val(A) op val(B). By induction hypothesis, $|\alpha_i| \leq u(A)$ and $|\beta_j| \leq u(B)$. If op $\in \{+, -\}$, the roots of $p_E(X)$ are bounded by

$$|\alpha_i \text{ op } \beta_j| \leq |\alpha_i| + |\beta_j| \leq u(A) + u(B) = u(E).$$

If op $= *$, the roots of $p_E(X)$ are bounded by

$$|\alpha_i \cdot \beta_j| \leq |\alpha_i| \cdot |\beta_j| \leq u(A) \cdot u(B) = u(E).$$

If $E = \sqrt[k]{A}$ and $p_A(X)$ as above, we set

$$p_E(X) = p_A(X^d) = \prod_{i=1}^{n} (X^d - \alpha_i) = \prod_{i=1}^{n} \prod_{\delta=0}^{d-1} (X - \zeta_d^{\delta} \sqrt[k]{\alpha_i}) \in \mathbb{Z}(X),$$

where ζ_d is a (complex) primitive dth root of unity. In this case

$$|\zeta_d^{\delta} \sqrt[k]{\alpha_i}| = \sqrt[k]{|\alpha_i|} \leq \sqrt[k]{u(A)} = u(E),$$

which concludes the induction step and thus the proof of our lemma. □

In Lemma 6 we constructed a polynomial $p_E(X) \in \mathbb{Z}(X)$ with root ξ. Since every integral polynomial having root ξ is divided by the minimal polynomial $m_\xi(X)$, see, e.g., [14] and [15], both $m_\xi(X)$ and $p_E(X)$ have leading coefficient 1, and the roots of $m_\xi(X)$ are among the roots of $p_E(X)$. Hence

$$m_\xi(X) = \prod_{j=1}^{deg(\xi)} (X - e_{i_j})$$

for some indices $i_1, \dots, i_{deg(\xi)}$. Since $m_\xi(X)$ is the minimal polynomial, none of the roots e_{i_j} is zero. By Lemma 6 all roots are bounded by $u(E)$. Without loss of generality $e_{i_1} = \xi$. Since $\prod_j e_{i_j} \in \mathbb{Z}\backslash\{0\}$ we have

$$|\xi| \geq \frac{|\xi|}{\left|\prod_{j=1}^{deg(\xi)} e_{i_j}\right|} = \frac{1}{\left|\prod_{j=2}^{deg(\xi)} e_{i_j}\right|} \geq \frac{1}{u(E)^{deg(\xi)-1}}.$$

This completes the proof of Theorem 1. □

Corollary 2 follows from Theorem 1 because $deg(\xi) \leq \prod_{i=1}^r k_i$.

For the proof of Corollary 3 we apply the bounds of Corollary 2 to the division-free expressions $U(E)$ and $L(E)$. Each radical operation of index k_i in E generates at most two radical operations of index k_i, each of which may appear in $U(E)$ as well as in $L(E)$. Thus $D(U(E)) \leq \prod_{i=1}^r k_i^2 = D(E)^2$ and $D(L(E)) \leq D(E)^2$. Since $\text{val}(E) = \text{val}(U(E))/\text{val}(L(E))$, and $u(U(E)) = u(E)$ and $u(L(E)) = l(E)$, we have

$$\left(l(E)u(E)^{D^2(E)-1}\right)^{-1} \leq |\xi| \leq u(E)l(E)^{D^2(E)-1}. \qquad \square$$

3. Comparison with Previous Work.

There is a rich body of literature on root separation bounds [5], [8], [12], [13], the work by Mignotte [12], [13] and Canny [5] being particularly relevant for this paper.

Mignotte [12] considers the identification of algebraic numbers, say α and β, given as algebraic expressions. Looking for a "numerical proof" of the identity of α and β he discusses bounds on $|\alpha - \beta|$. In our case $\beta = 0$. Mignotte defines functions measuring the *size* of algebraic numbers. The definition of a size-function includes inequalities that relate the size of the sum and the product of algebraic numbers to the sizes of the operands. Since the definition of a size-function does not involve inequalities for square root operation and quotient, it is not obvious how to get an easily computable separation bound based on such size-functions.

The recursive rules given in Table 3 of quantities $M(E)$ and $Deg(E)$ for an expression E are in the spirit of Section 5 of [12] and based on the measure of an algebraic number, see Section 4.3 in [13]. We have $|\text{val}(E)| \geq M(E)^{-1}$ for all expressions E. We call this the *degree-measure bound*. For the expression $E = (2^{2^k} + 1)^{2^{-k}} - 2$ considered in the Introduction the degree-measure bound is $8^{-2^k - 1/3}$ which is slightly worse than ours. We now show that this is not a coincidence.

We introduce a new quantity $S(E)$ to make the relation between the degree-measure bound and our bound more evident. $S(E)$ is defined in Table 4. It is straightforward to see that $u(E) = u(U(E)) \leq S(E)$ and $l(E) = u(L(E)) \leq S(E)$.

Table 3. Inductive rules for degree-measure bound.

E	$M(E)$	$Deg(E)$
Integer n	$\lvert n\rvert$	1
$E_1 \pm E_2$	$2^{Deg(E_1)\cdot Deg(E_2)} \cdot M(E_1)^{Deg(E_2)} \cdot M(E_2)^{Deg(E_1)}$	$Deg(E_1)\cdot Deg(E_2)$
$E_1 \cdot E_2$	$M(E_1)^{Deg(E_2)} \cdot M(E_2)^{Deg(E_1)}$	$Deg(E_1)\cdot Deg(E_2)$
E_1/E_2	$M(E_1)^{Deg(E_2)} \cdot M(E_2)^{Deg(E_1)}$	$Deg(E_1)\cdot Deg(E_2)$
$\sqrt[k]{E_1}$	$M(E_1)$	$k \cdot Deg(E_1)$

LEMMA 7. *For every expression E we have $S(E)^{Deg(E)} = M(E)$.*

PROOF. The proof is by induction on the structure of E. If E is an integer constant a we have $S(E)^{Deg(E)} = |a| = M(E)$. If $E = E_1 \pm E_2$, then we have

$$\begin{aligned} S(E)^{Deg(E)} &= 2^{Deg(E)} S(E_1)^{Deg(E)} S(E_2)^{Deg(E)} \\ &= 2^{Deg(E)} \left(S(E_1)^{Deg(E_1)}\right)^{Deg(E_2)} \left(S(E_2)^{Deg(E_2)}\right)^{Deg(E_1)} \\ &= 2^{Deg(E)} M(E_1)^{Deg(E_2)} M(E_2)^{Deg(E_1)} \\ &= M(E). \end{aligned}$$

If $E = E_1 \cdot E_2$ or $E = E_1/E_2$ the computation is similar. If $E = \sqrt[k]{E_1}$ we have

$$S(E)^{Deg(E)} = \sqrt[k]{S(E_1)}^{(k\cdot Deg(E_1))} = S(E_1)^{Deg(E_1)} = M(E_1) = M(E). \qquad \square$$

Next we show that our separation bound is never worse than the degree-measure bound for a division-free expression E.

LEMMA 8.

$$M(E) \geq u(E)^{D(E)-1}.$$

PROOF. Since $Deg(E) \geq D(E)$, we have $M(E) = S(E)^{Deg(E)} \geq u(E)^{Deg(E)} \geq u(E)^{D(E)-1}$. $\square$

Note that $Deg(E)$ coincides with the degree of the polynomial $p_E(X)$ constructed in the proof of Theorem 1. This is not surprising since the polynomial $p_E(X)$ also appears implicitly in the degree-measure bound construction.

Table 4. Inductive rules for $S(E)$.

E	$S(E)$
Integer n	$\lvert n\rvert$
$E_1 \pm E_2$	$2 \cdot S(E_1) \cdot S(E_2)$
$E_1 \cdot E_2$	$S(E_1) \cdot S(E_2)$
E_1/E_2	$S(E_1) \cdot S(E_2)$
$\sqrt[k]{E_1}$	$\sqrt[k]{S(E_1)}$

Canny [5] considers systems of n polynomial equations in n unknowns. He shows that if $a_1, \ldots, a_n$ is any isolated solution to such a system, then

$$a_i = 0 \quad \text{or} \quad |a_i| \geq (3dc)^{-n \cdot d^n}$$

for all i, where d is the maximal degree of any polynomial in the system and c is the maximal coefficient appearing in any of the polynomials. Hong [8] had previously proven a similar bound for tridiagonal systems.

There are many ways to transform expression dags into polynomial systems, e.g., one may either introduce a variable for each node of the dag and then relate the input and output variables of each node by an equation, or one may introduce a variable for each square root and each division. For the expression $E = (2^{2^k} + 1)^{2^{-k}} - 2$ considered in the introduction the two methods yield

$$\begin{array}{rl} x_1 &= 2, \\ x_2 &= x_1^2, \\ &\vdots \\ x_k &= x_{k-1}^2 + 1, \\ y_k^2 &= x_k, \\ &\vdots \\ y_1^2 &= y_2, \\ y &= y_1 - 2, \end{array} \qquad \text{and} \qquad \begin{array}{rl} x_k &= 2^{2^k} + 1, \\ y_k^2 &= x_k, \\ &\vdots \\ y_1^2 &= y_2, \\ y &= y_1 - 2, \end{array}$$

respectively, i.e., we have $n = 2k+1$, $d = 2$, and $c = 2$ in the former case, and $n = k+2$, $d = 2$, and $c = 2^{2^k} + 1$ in the latter case. The bounds are therefore $12^{-(2k+1)\cdot 2^{2k+1}}$ and $(6 \cdot (2^{2^k} + 1))^{-(k+2)\cdot 2^{k+2}}$, respectively. Try $k = 3$ to see the difference between these bounds and our bound.

We show next that our bound is never worse than the bound derived from Canny's result for division-free expressions involving square roots. Let $d \geq 2$ and c be integers and consider a system

$$x_i^2 = p_i(x_1, \ldots, x_{i-1}), \qquad 1 \leq i \leq r,$$

where each p_i is a polynomial of degree at most d whose maximal coefficient is bounded by c. By writing

$$x_i = \sqrt{p_i(x_1, \ldots, x_{i-1})}$$

the system turns into a dag with r square root nodes. Let u_i be $u(x_i)$ as computed by the rules given in the Introduction. If $(a_1, \ldots, a_r)$ is any solution of this system, then

$$a_i = 0 \quad \text{or} \quad |a_i| \geq (3dc)^{-i \cdot d^i}$$

according to Canny and

$$a_i = 0 \quad \text{or} \quad |a_i| \geq u_i^{1-2^i} \geq u_i^{-2^i}$$

according to Theorem 1.

LEMMA 9. *$u_i^{-2^i} \geq (3dc)^{-i \cdot d^i}$ for all i.*

PROOF. It suffices to prove that

$$u_i \leq (3dc)^{i \cdot (d/2)^i}$$

for all i. We use induction on i. The first equation is $x_1^2 = c'$ for some $c' \leq c$. Thus, $u_1 = \sqrt{c'}$ and the inequality clearly holds for $i = 1$.

For the induction step we have

$$u_{i+1} \leq \left(c \cdot \sum_{l_1 + \cdots + l_i \leq d} u_1^{l_1} \ldots u_i^{l_i} \right)^{1/2}.$$

We now distinguish cases. For $d = 2$ we have $u_l \leq (6c)^l$ for $l \leq i$ by induction hypothesis and hence

$$\begin{aligned}
u_{i+1} &\leq c^{1/2} \cdot \left(1 + \sum_{l=1}^{i} u_l \cdot \left(1 + \sum_{j=1}^{l} u_j \right) \right)^{1/2} \\
&\leq c^{1/2} \cdot \left(1 + \sum_{l=1}^{i} (6c)^l \cdot \sum_{j=0}^{l} (6c)^j \right)^{1/2} \\
&\leq c^{1/2} \cdot \left(1 + \sum_{l=1}^{i} (6c)^l \cdot \left((6c)^{l+1} - 1 \right) / (6c - 1) \right)^{1/2} \\
&\leq c^{1/2} \cdot \left(\sum_{l=0}^{i} \left((6c)^{2l+1} - (6c)^l \right) / (6c - 1) \right)^{1/2} \\
&\leq \left(\sum_{l=0}^{i} ((6c)^{2l+1} \right)^{1/2} \\
&\leq \left((6c)^{2i+2} \right)^{1/2} \\
&= (6c)^{i+1},
\end{aligned}$$

since $c/(6c - 1) \leq 1$ for $c \geq 1$.

For $d \geq 3$ we have $u_l \leq (3dc)^{l \cdot (d/2)^l}$ for $l \leq i$ and hence

$$\begin{aligned}
u_{i+1} &\leq (c \cdot (d+1)^i \cdot u_i^d)^{1/2} \\
&\leq (c \cdot (d+1)^i \cdot (3dc)^{i \cdot (d/2)^i \cdot d})^{1/2} \\
&\leq \left(c^{1/2} \cdot (d+1)^{i/2} / (3dc)^{(d/2)^{i+1}} \right) \cdot (3dc)^{(i+1) \cdot (d/2)^{i+1}} \\
&\leq (3dc)^{(i+1) \cdot (d/2)^{i+1}}
\end{aligned}$$

since $i/2 \leq (d/2)^{i+1}$ for $d \geq 3$ and $i \geq 1$. □

4. Applications. Our motivation for considering separation bounds is their use for testing the sign of real-valued expressions. Let E be a real-valued expression and let $\text{sep}(E)$ be a separation bound for E. We can then compute the sign of E as follows. We compute an interval I_ε of length ε such that $0 < \varepsilon < \text{sep}(E)$ and I_ε contains E. If 0 is inside I_ε, the value of E is smaller than $\text{sep}(E)$ and hence E is zero. Otherwise, the sign of E is equal to the sign of the numbers in I_ε.

LEDA [11] provides a number type `real` that uses the concept of separation bounds for testing the sign of an arithmetic expression over the operations $+, -, *, /, \sqrt{\ }$. Dubé and Yap [6] provide a similar number type called `bigExpression`. The procedure used in `bigExpression` for computing a separation bound is based on a bound, called *degree-length bound* in [6], which is almost identical to the degree-measure bound described in Section 3. Since their rules for product and quotient involve an additional factor of $2^{\min(Deg(E_1),Deg(E_2))}$, the resulting bounds for expressions over operators $+, -, *, /, \sqrt{\ }$, and integral operands are even weaker than the degree-measure bounds.

We implemented three procedures for computing separation bounds that use the bound of Theorem 1, the degree-measure bound, and a bound based on Canny's result, respectively. We call the bound based on Canny's result the *polynomial system bound*.

4.1. *Actual Computation of Separation Bounds*. Theorem 1 cannot be applied directly since it evolves quantities $u(E)$ and $l(E)$ that are in general nonrational algebraic numbers. We propose two methods to compute slightly weaker separation bounds, firstly computing with floating point numbers with rounding toward larger values and secondly maintaining integral logarithms of $u(E)$ and $l(E)$.

For the first method we assume the IEEE standard 754 for binary floating point arithmetic [1]. The IEEE standard 754 requires that the result of the floating point operations $+, -, *, /,$ and $\sqrt{\ }$ are exactly rounded according to the chosen rounding mode. Moreover, the standard requires the support of rounding toward infinity, i.e., to larger values. Using this rounding mode we can compute upper bounds for $u(E)$ and $l(E)$ with floating point arithmetic.

Changing the rounding mode is expensive on some systems. As an alternative one can simply evaluate $u(E)$ and $l(E)$ in double precision floating point arithmetic and multiply each intermediate result by $(1 + 2^{-52})$. Note that underflow does not occur here since we always have $u(E) \geq 1$ whenever $u(E)$ is nonzero. In case of overflow, the resulting value of $u(E)$ will be infinity, which is a correct (but meaningless) bound.

In the second method we recursively compute integers $u'(E) \geq \log_2(u(E))$ and $l'(E) \geq \log_2(l(E))$ as shown in Table 5. The recursive rules for $u'(E)$ and $l'(E)$ do not

Table 5. Inductive rules for easily computable $u'(E)$ and $l'(E)$.

E	$u'(E)$	$l'(E)$
Integer n	$\lceil \log_2(n) \rceil$	0
$E_1 \pm E_2$	$1 + \max(u'(E_1) + l'(E_2), l'(E_1) + u'(E_2))$	$l'(E_1) + l'(E_2)$
$E_1 \cdot E_2$	$u'(E_1) + u'(E_2)$	$l'(E_1) + l'(E_2)$
E_1/E_2	$u'(E_1) + l'(E_2)$	$l'(E_1) + u'(E_2)$
$\sqrt[k]{E_1}$	$\lceil u'(E_1)/k \rceil$	$\lceil l'(E_1)/k \rceil$

evolve radical operations. All numbers arising in the computation of $u'(E)$ and $l'(E)$ are integers, and fairly small in practice. Hence rounding errors are not an issue for computing $u'(E)$ and $l'(E)$.

We implemented the second method and compared the results with the polynomial system bound and the degree-measure bound. In our implementation the degree-measure bound is maintained logarithmically as well, as suggested in [12]. Furthermore, instead of recursively computing $Deg(E)$ we use the estimate $2^{r(E)}$ to bound the algebraic degree of expression E, where $r(E)$ is the number of square root nodes in the expression dag E. In the polynomial system bound we maintain bounds on the degree and coefficient size of the implicitly generated system of polynomial equations.

4.2. *Application to Voronoi Diagram Computation.* The following incircle tests arise in the computation of Voronoi diagrams of line segments and points. Let v be the Voronoi vertex having minimal distance to sites s_1, s_2, and s_3 and let s be another site. Is s inside the Voronoi circle of v, i.e., is it nearer to v than the sites s_1, s_2, s_3? In the implementation of the incircle test we compare the squared distance of the sites s_1, s_2, and s_3 to v with the squared distance of s to v.

For example, let v be given by two segments s_1, s_2 and a point p and let s be a segment s_3. Let l_i: $a_i x + b_i y + c_i = 0$ be the supporting line of segment s_i. In [2] it is shown that for $p = (0, 0)$ the incircle test includes a sign test of the form $E < 0$ for

$$E = (x_v^2 + y_v^2)(a_3^2 + b_3^2) - (a_3 x_v + b_3 y_v + c_3 z_v)^2,$$

where the coordinates $(x_v/z_v, y_v/z_v)$ of vertex v are

$$\begin{aligned} x_v &= a_1c_2 + a_2c_1 \pm \sqrt{2c_1c_2(\sqrt{N} + (a_1a_2 - b_1b_2))}, \\ y_v &= b_1c_2 + b_2c_1 \pm \sqrt{2c_1c_2(\sqrt{N} - (a_1a_2 - b_1b_2))}, \\ z_v &= \sqrt{N} - (a_1a_2 + b_1b_2), \end{aligned}$$

and $N = (a_1^2 + b_1^2)(a_2^2 + b_2^2)$.

There are different types of incircle tests that correspond to the different types of sites defining v and s. s can have type p (point) or l (line segment) and v can have type ppp, ppl, llp, or lll where a symbol p stands for a point among the sites s_i and a symbol l stands for a line segment among the sites s_i. For example, the incircle test above has type llp-l.

We assume that input points have b bit integral Cartesian coordinates and line segments are given by two input points. Our main goal is to find the exact dependence of the separation bound of a test expression on the maximal bit size b of the integer coordinates. Note that for any of the methods discussed here the logarithm of the separation bound for expression E is a linear function $s(b) = s_E(b)$ of b. We split $s(b)$ into $s(b) = s_b b + s_c$ and compute the values s_b and s_c separately. s_b and s_c can be computed recursively by formulas that are analogous to the formulas given before. For the various incircle tests, we get the results shown in Table 6.

Compared with the bounds based on the results of Canny and Mignotte, our new separation bounds lead to a considerable improvement. As proven in Section 3, the

Table 6. Comparison of separation bounds for incircle tests.

Incircle test	Our bound	Polynomial system bound	Degree-measure bound
`ppp-p`	$0{\cdot}b+0$	$4{\cdot}b+17$	$92{\cdot}b+39$
`ppp-l`	$0{\cdot}b+0$	$8{\cdot}b+42$	$202{\cdot}b+91$
`ppl-p`	$6{\cdot}b+14$	$64{\cdot}b+312$	$2{,}032{\cdot}b+1{,}034$
`ppl-l`	$12{\cdot}b+27$	$96{\cdot}b+680$	$8{,}864{\cdot}b+4{,}586$
`llp-p`	$28{\cdot}b+84$	$384{\cdot}b+1728$	$156{,}416{\cdot}b+82{,}584$
`llp-l`	$56{\cdot}b+161$	$512{\cdot}b+3840$	$1{,}307{,}136{\cdot}b+697{,}736$
`lll-p`	$70{\cdot}b+147$	$512{\cdot}b+3456$	$14{,}188{,}928{\cdot}b+6{,}165{,}768$
`lll-l`	$84{\cdot}b+168$	$640{\cdot}b+4352$	$11{,}404{,}160{\cdot}b+4{,}762{,}120$

polynomial system bounds are always weaker than ours but, at least in these examples, are still usable for the sign computation. For the last four tests the degree-measure bounds are much too weak to be of any practical use. The reader should realize that computing just one single square root operation up to a million binary digits is not feasible in a reasonable time on most computers.

Comparing Table 6 with the results of [2] which were computed by hand, we see that we get the same asymptotic bounds for the first four tests and slightly weaker bounds for the last four tests. Note that the test expressions for ppp-p and ppp-l are integral and hence 1 is an obvious separation bound in these cases. Only our method achieves this bound. On the other hand, there are several reasons why our results are not always optimal. In the test expression E for llp-l (and also for llp-p), we have three different square root operations. However, the two nested roots $\sqrt{F}$ and $\sqrt{G}$ that appear in E are algebraically dependent over the rationals and, in particular, $\sqrt{F}\sqrt{G}$ is an integer. Hence two of the three square roots that appear in E are algebraically dependent and hence E has in fact algebraic degree 2^2 at most and not 2^3 as is implicitly assumed in the computation of our bound.

References

[1] IEEE Standard 754-1985 for Binary Floating-Point Arithmetic. Reprinted in *SIGPLAN*, 22(2):9–25, 1987.

[2] C. Burnikel. Exact Computation of Voronoi Diagrams and Line Segment Intersections. Ph.D. thesis, Universität des Saarlandes, 1996.

[3] C. Burnikel, K. Mehlhorn, and S. Schirra. How to Compute the Voronoi Diagram of Line Segments: Theoretical and Experimental Results. In *Proceedings of ESA* '94, pages 227–239. LNCS 855, Springer-Verlag, Berlin, 1994.

[4] C. Burnikel, K. Mehlhorn, and S. Schirra. The LEDA Class Real Number. Technical Report MPI-I-96-1-001, Max-Planck-Institut für Informatik, Saarbrücken, January 1996. see `http://data.mpi-sb.mpg.de/internet/reports.nsf/NumberView/1996-1-001`.

[5] J.F. Canny. *The Complexity of Robot Motion Planning*. The MIT Press, Cambridge, MA, 1987.

[6] T. Dubé and C.K. Yap. A Basis for Implementing Exact Geometric Algorithms. Manuscript, October 1993.

[7] E. Hecke, *Vorlesungen über die Theorie der Algebraischen Zahlen*, 2nd edition. Chelsea, New York, 1970.

[8] J.W. Hong. Proving by Example and Gap Theorem. In *Proceedings of the* 28*th Symposium on the Foundations of Computer Science* (*FOCS* '87), 1987.

[9] G. Liotta, F.P. Preparata, and R. Tamassia. Robust Proximity Queries in Implicit Voronoi Diagrams. Technical Report RI 02912-1910, Center for Geometric Computing, Department of Computer Science, Brown University, Providence, RI, May 1996.

[10] R. Loos. Computing in Algebraic Extensions. In B. Buchberger, G.E. Collins, and R. Loos, editors, *Computer Algebra*, pages 173–187. Springer-Verlag, Berlin, 1982.

[11] K. Mehlhorn and S. Näher. *LEDA*: *A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, 1999, see `http://www.mpi-sb.mpg.de/LEDA/leda.html`.

[12] M. Mignotte. Identification of Algebraic Numbers. *Journal of Algorithms*, 3(3), September 1982.

[13] M. Mignotte. *Mathematics for Computer Algebra*. Springer-Verlag, Berlin, 1992.

[14] B.L. van der Waerden. *Algebra*, volumes 1 & 2. Ungar, New York, 1970.

[15] C.K. Yap. Towards Exact Geometric Computation. In *Proceedings of CCCG* ‘93, pages 405–419, 1993.

[16] C.K. Yap. *Fundamental Problems in Algorithmic Algebra*. Princeton University Press, Princeton, NJ (to appear).

[17] C. K. Yap and T. Dubé. The Exact Computation Paradigm. In D.Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*, 2nd edition, pages 452–492. World Scientific, Singapore, 1995.