Root finding with threshold circuits

Emil Jeřábek*

Institute of Mathematics of the Academy of Sciences
Žitná 25, 115 67 Praha 1, Czech Republic, email: jerabek@math.cas.cz

August 27, 2012

Abstract

We show that for any constant d, complex roots of degree d univariate rational (or Gaussian rational) polynomials—given by a list of coefficients in binary—can be computed to a given accuracy by a uniform TC^0 algorithm (a uniform family of constant-depth polynomial-size threshold circuits). The basic idea is to compute the inverse function of the polynomial by a power series. We also discuss an application to the theory VTC^0 of bounded arithmetic.

1 Introduction

The complexity class TC^0 was originally defined by Hajnal et al. [15] in the nonuniform setting, as the class of problems recognizable by a family of polynomial-size constant-depth circuits with majority gates. It was implicitly studied before by Parberry and Schnitger [27], who consider various models of computation using thresholds (threshold circuits, Boltzmann machines, threshold RAM, threshold Turing machines). The importance of the class follows already from the work of Chandra, Stockmayer, and Vishkin [7], who show (in today's terminology) the TC^0-completeness of several basic problems (integer multiplication, iterated addition, sorting) under AC^0 reductions. Barrington, Immerman, and Straubing [4] establish that there is a robust notion of fully uniform TC^0. (We will use TC^0 to denote this uniform TC^0, unless stated otherwise.)

We can regard TC^0 as the natural complexity class of elementary arithmetical operations: integer multiplication is TC^0-complete, whereas addition, subtraction, and ordering are in $\mathrm{AC}^0 \subseteq \mathrm{TC}^0$. The exact complexity of division took some time to settle. Wallace [33] constructed division circuits of depth $O((\log n)^2)$ and bounded fan-in (i.e., NC^2). Reif [29] improved this bound to $O(\log n \log\log n)$. Beame, Cook, and Hoover [5] proved that division, iterated multiplication, and exponentiation (with exponent given in unary) are TC^0-reducible to each other, and constructed P-uniform TC^0 circuits for these problems. Chiu, Davida, and

*Supported by grant IAA100190902 of GA AV ČR, project 1M0545 of MŠMT ČR, and RVO: 67985840.

Litow [8] exhibited logspace-uniform TC^0 circuits for division, showing in particular that division is computable in L. Finally, Hesse, Allender, and Barrington [16] proved that division (and iterated multiplication) is in uniform TC^0.

Using these results, other related problems can be shown to be computable in TC^0, for example polynomial division, iterated multiplication, and interpolation. In particular, using iterated addition and multiplication of rationals, it is possible to approximate in TC^0 functions presented by sufficiently nice power series, such as log, exp, $x^{1/k}$, and trigonometric functions, see e.g. Reif [29], Reif and Tate [30], Maciel and Thérien [20], and Hesse et al. [16].

Numerical computation of roots of polynomials is one of the oldest problems in mathematics, and countless algorithms have been devised to solve it, both sequential and parallel. The most popular methods are based on iterative techniques that successively derive closer and closer approximations to a root (or, sometimes, to all the roots simultaneously) starting from a suitable initial approximation. Apart from the prototypical Newton–Raphson iteration, there are for instance Laguerre's method [28, §9.5], Brent's method [28, §10.3], the Durand–Kerner method [13, 19], the Jenkins–Traub algorithm [17], and many others. One can also reduce root finding to matrix eigenvalue computation, for which there are iterative methods such as the QR algorithm [14]. Another class of root-finding algorithms are divide-and-conquer approaches: the basic idea is to recursively factorize the polynomial by identifying a suitable contour (typically, a circle) splitting the set of roots roughly in half, and recovering coefficients of the factor whose roots fall inside the contour from the residue theorem by numerical integration. Algorithms of this kind include Pan [24], Ben-Or et al. [6], Neff [22], Neff and Reif [23], and Pan [25], see Pan [26] for an overview. These algorithms place root finding in NC: for example, the algorithm of [25] can find n-bit approximations to all roots of a polynomial of degree $d \le n$ in time $O((\log n)^2(\log d)^3)$ using $O(nd^2(\log\log n)/(\log d)^2)$ processors on an EREW PRAM. (More specifically, Allender [3] mentions that root finding is known to be in the #L hierarchy, but not known to be in GapL.)

The purpose of this paper is to demonstrate that in the case of constant-degree polynomials, we can push the complexity of root finding down to uniform TC^0 (i.e., constant time on polynomially many processors on a TRAM, in terms of parallel complexity), as in the case of elementary arithmetical operations. (This is clearly optimal: already locating the unique root of a linear polynomial amounts to division, which is TC^0-hard.) As a corollary, the binary expansion of any algebraic constant can be computed in uniform TC^0 when given the bit position in unary. Our primary interest is theoretical, we seek to investigate the power of the complexity class TC^0; we do not expect our algorithm to be competitive with established methods in practice, and we did not make any effort to optimize parameters of the algorithm.

The basic idea of the algorithm is to express the inverse function of the polynomial by a power series, whose partial sums can be computed in TC^0 using the results of Hesse et al. [16]. We need to ensure that coefficients of the series are TC^0-computable, we need bounds on the radius of convergence and convergence rate of the series, and we need to find a point in whose image to put the centre of the series so that the disk of convergence includes the origin. Doing the latter directly is in fact not much easier than approximating the root in the first place, so we instead construct a suitable polynomial-size set of sample points, and we invert the polynomial at each one of them in parallel.

We formulated our main result in terms of computational complexity, but our original motivation comes from logic (proof complexity). The bounded arithmetical theory VTC^0 (see Cook and Nguyen [12]), whose provably total computable functions are the TC^0 functions, can define addition, multiplication, and ordering on binary integers, and it proves that these operations obey the basic identities making it a discretely ordered ring. The question is which other properties of the basic arithmetical operations are provable in the theory, and in particular, whether it can prove induction (on binary integers) for some class of formulas. Now, it follows easily from known algebraic characterizations of induction for open formulas in the language of ordered rings (*IOpen*, see Shepherdson [32]) and from the witnessing theorem for VTC^0 that VTC^0 proves *IOpen* if and only if for each d there is a TC^0 root-finding algorithm for degree d polynomials whose soundness is provable in VTC^0. Our result thus establishes the computational prerequisites for proving open induction in VTC^0, leaving aside the problem of formalizing the algorithm in the theory. Since the soundness of the algorithm can be expressed as a universal sentence, we can also reformulate this result as follows: the theory $VTC^0 + \mathrm{Th}_{\forall\Sigma^B_0}(\mathbb{N})$ proves *IOpen*.

The paper is organized as follows. In Section 2 we provide some background in the relevant parts of complexity theory and complex analysis. Section 3 contains material on inverting polynomials with power series. Section 4 presents our main result, a TC^0 root-finding algorithm. Finally, in Section 5 we discuss the connection to bounded arithmetic.

2 Preliminaries

A language L is in *nonuniform* TC^0 if there is a sequence of circuits $C_n\colon \{0,1\}^n \to \{0,1\}$ consisting of unbounded fan-in majority and negation gates such that C_n computes the characteristic function of L on strings of length n, and C_n has size at most n^c and depth c for some constant c.

L is in (*uniform*) TC^0, if the sequence $\{C_n : n \in \omega\}$ is additionally DLOGTIME-uniform (U_D-uniform in the terminology of Ruzzo [31]): i.e., we can enumerate the gates in the circuit by numbers $i < n^{O(1)}$ in such a way that one can check the type of gate i and whether gate i is an input of gate j by a deterministic Turing machine in time $O(\log n)$, given n, i, j in binary. There are other equivalent characterizations of TC^0. For one, it coincides with languages recognizable by a threshold Turing machine [27] in time $O(\log n)$ with $O(1)$ thresholds [2]. Another important characterization is in terms of descriptive complexity. We can represent a string $x \in \{0,1\}^n$ by the first-order structure $\langle \{0,\dots,n-1\}, <, \mathrm{bit}, X\rangle$, where X is a unary predicate encoding the bits of x. Then a language is in TC^0 iff its corresponding class of structures is definable by a sentence of FOM (first-order logic with majority quantifiers). We refer the reader to [4] for more background on uniformity of TC^0.

In some cases it may be more convenient to consider languages in a non-binary alphabet Σ. The definition of TC^0 can be adapted by adjusting the input alphabet of a threshold Turing machine, or by considering more predicates in the descriptive complexity setting. In the original definition using threshold circuits, the same can be accomplished by encoding each symbol of Σ with a binary substring of fixed length. We can also define TC^0 predicates with more than one input in the obvious way.

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is computable in TC^0 if the length of its output is polynomially bounded in the length of its input, and its bitgraph is a TC^0 predicate. (The bitgraph of f is a binary predicate $b(x,i)$ which holds iff the ith bit of $f(x)$ is 1.) In terms of the original definition, this amounts to allowing circuits $C_n\colon \{0,1\}^n \to \{0,1\}^{m(n)}$, where $m(n) = n^{O(1)}$. TC^0 functions are closed under composition, and under "parallel execution": if f is a TC^0 function, its aggregate function $g(\langle x_0,\dots,x_{m-1}\rangle) = \langle f(x_0),\dots,f(x_{m-1})\rangle$ is also in TC^0. We note in this regard that TC^0 functions can do basic processing of lists

$$x_0, x_1, \dots, x_{m-1}$$

where "," is a separator character. Using the fact that TC^0 can count commas (and other symbols), we can for instance extract the ith element from the list, convert the list to and from a representation where each element is padded to some fixed length with blanks, or sort the list according to a given TC^0 comparison predicate.

We will refrain from presenting TC^0 functions in one of the formalisms suggested by the definitions above: we will give informal algorithms, generally consisting of a constant number of simple steps or TC^0 building blocks, sometimes forking into polynomially many parallel threads. The reader should have no difficulty convincing herself that our algorithms are indeed in TC^0.

We will work with numbers of various kinds. Integers will be represented in binary as usual, unless stated otherwise. As we already mentioned in the introduction, elementary arithmetical operations on integers are TC^0 functions: this includes addition, subtraction, ordering, multiplication, division with remainder, exponentiation (with unary exponents), iterated addition, iterated multiplication, and square root approximation. Here, iterated addition is the function $\langle x_0,\dots,x_{m-1}\rangle \mapsto \sum_{i<m} x_i$, and similarly for multiplication. Notice that using iterated multiplication, we can also compute factorials and binomial or multinomial coefficients of unary arguments. Base conversion is also in TC^0.

Rational numbers will be represented as pairs of integers, indicating fractions. We cannot assume fractions to be reduced, since integer gcd is not known to be TC^0-computable. Using integer division, we can convert a fraction to its binary expansion with a given accuracy (the opposite conversion is trivial). Rational arithmetic is reducible to integer arithmetic in the obvious way, hence rational addition, subtraction, ordering, multiplication, division, exponentiation (with unary integer exponents), iterated addition, iterated multiplication, and square root approximation are in TC^0.

In lieu of complex numbers, we will compute with Gaussian rationals (elements of the field $\mathbb{Q}(i)$), represented as pairs of rationals $a + ib$. By reduction to rational arithmetic, we can see that addition, subtraction, complex conjugation, norm square, norm approximation, multiplication, division, and iterated addition of Gaussian rationals are in TC^0. Using the binomial theorem, exponentiation with unary integer exponents is also in TC^0. (In fact, iterated multiplication of Gaussian rationals is in TC^0 using conversion to polar coordinates, but we will not need this.)

We will need some tools from complex analysis. We refer the reader to Ahlfors [1] or Conway [10] for background, however, we review here some basic facts to fix the notation. A function $f\colon U \to \mathbb{C}$, where $U \subseteq \mathbb{C}$ is open, is *holomorphic* (or *analytic*) in U if $f'(a) = \lim_{z\to a}(f(z) - f(a))/(z - a)$ exists for every $a \in U$. The set of all functions holomorphic in U is denoted $H(U)$. Let $B(a, r) := \{z : |z - a| < r\}$ and $\overline{B}(a, r) := \{z : |z - a| \leq r\}$. If f is holomorphic in the open disk $B(a, R)$, it can be expressed by a *power series*

$$f(z) = \sum_{n=0}^{\infty} c_n(z - a)^n$$

on $B(a, R)$. More generally, if f is holomorphic in the annulus $A = B(a, R) \smallsetminus \overline{B}(a, r)$, $0 \leq r < R \leq \infty$, it can be written in A as a *Laurent series*

$$f(z) = \sum_{n=-\infty}^{+\infty} c_n(z - a)^n.$$

We denote the coefficients of the series by $[(z - a)^n]f := c_n$. (Other variables may be used instead of z when convenient.) The *residue* of f at a is $\operatorname{Res}(f, a) := [(z-a)^{-1}]f$. When $a = 0$, we write just $[z^n]f$ and $\operatorname{Res}(f)$, respectively. The coefficients of a Laurent series are given by *Cauchy's integral formula*:

$$[(z - a)^n]f = \frac{1}{2\pi i}\int_\gamma \frac{f(z)}{(z - a)^{n+1}}\, dz,$$

where γ is any closed curve in A whose index with respect to a is 1 (such as the circle $\gamma(t) = a + \varrho e^{2\pi i t}$, $r < \varrho < R$). The *identity theorem* states that if f, g are holomorphic in a region (i.e., connected open set) U and coincide on a set $X \subseteq U$ which has a limit point in U, then $f = g$. The *open mapping theorem* states that a nonconstant function f holomorphic in a region is an open mapping (i.e., maps open sets to open sets).

If $X \subseteq \mathbb{C}$ and $a \in \mathbb{C}$, we put $\operatorname{dist}(a, X) = \inf\{|z - a| : z \in X\}$.

We will also need some easy facts on zeros of polynomials. Let $f \in \mathbb{C}[x]$ be a degree d polynomial, and write $f(x) = \sum_{j=0}^{d} a_j x^j$. *Cauchy's bound* [34, L. 6.2.7] states that every zero α of f satisfies

$$\frac{|a_0|}{|a_0| + \max\limits_{0<j\leq d}|a_j|} \leq |\alpha| \leq 1 + \max_{j<d}\frac{|a_j|}{|a_d|}.$$

Let $f, g \in (\mathbb{Q}(i))[x]$ be two polynomials of degrees d, e (resp.), and assume $f(\alpha) = g(\beta) = 0$, $\alpha \neq \beta$. If $f, g \in (\mathbb{Z}[i])[x]$, we have

$$(*) \qquad |\alpha - \beta| \geq \frac{1}{(2^{d+1}\|f\|_\infty)^e\|g\|_2^d},$$

where $\|f\|_p$ denotes the L_p-norm of the vector of coefficients of f [34, §6.8]. In general, we can apply $(*)$ to the polynomials rf and sg, where r is the product of all denominators appearing among the coefficients of f, and similarly for s. If we represent f and g by the lists of their coefficients, which are in turn represented by quadruples of binary integers as detailed above, we obtain easily the following root separation bound:

Lemma 2.1 *For each $j = 0, 1$, let $f_j \in (\mathbb{Q}(i))[x]$ have degree d_j and total bit size n_j, and assume $f_j(\alpha_j) = 0$. If $\alpha_0 \neq \alpha_1$, then*

$$|\alpha_0 - \alpha_1| \geq 2^{-(d_1 n_0 + d_0 n_1)} \geq 2^{-n_0 n_1}.$$

3 Inverting polynomials

As already mentioned in the introduction, the main strategy of our algorithm will be to approximate a power series computing the inverse function of the given polynomial f. In this section, we establish the properties of such series needed to make the algorithm work.

The basic fact we rely on is that holomorphic functions with nonvanishing derivative are locally invertible: i.e., if $f \in H(U)$ and $a \in U$ is such that $f'(a) \neq 0$, there exist open neighbourhoods $a \in U_0 \subseteq U$ and $f(a) \in V_0$ such that f is a homeomorphism of U_0 onto V_0, and the inverse function $g = (f \restriction U_0)^{-1}$ is holomorphic in V_0. In particular, g is computable by a power series in a neighbourhood of $f(a)$.

Notice that local inverses of holomorphic functions are automatically two-sided: if $f \in H(U)$, $g \in H(V)$, $a \in U$, $b \in V$, $g(b) = a$, and $f(g(z)) = z$ in a neighbourhood of b, then $g(f(z)) = z$ in a neighbourhood of a.

The coefficients of the power series of an inverse of a holomorphic function are given by the *Lagrange inversion formula* [9, §3.8, Thm. A]:

Fact 3.1 *Let $f \in H(U)$, $g \in H(V)$, $f \circ g = \mathrm{id}_V$, $a = g(b) \in U$, $b = f(a) \in V$, $n > 0$. Then*

$$[(w-b)^n]g(w) = \frac{1}{n} \operatorname{Res}\left(\frac{1}{(f(z)-b)^n}, a\right).$$

We can make the formula even more explicit as follows. First, the composition of two power series is given by *Faà di Bruno's formula* [9, §3.4, Thm. A], which we formulate only for $a = b = 0$ for simplicity:

Fact 3.2 *Let $f \in H(U)$, $g \in H(V)$, $g(0) = 0 \in U$, $f(0) = 0 \in V$, $n \geq 0$. Then*

$$[z^n](g \circ f) = \sum_{\sum_{j=1}^{\infty} j m_j = n} \binom{\sum_j m_j}{m_1, m_2, \dots} [w^{\sum_j m_j}]g \prod_{j=1}^{\infty} \left([z^j]f\right)^{m_j}.$$

Note that here and below, the outer sum is finite, and the product has only finitely many terms different from 1, hence the right-hand side is well-defined without extra assumptions on convergence. We can now expand the residue in Fact 3.1 to obtain the following version of Lagrange inversion formula, which only refers to the coefficients of f [9, §3.8, Thm. E]:

Proposition 3.3 *Let $f \in H(U)$, $g \in H(V)$, $f \circ g = \mathrm{id}_V$, $a = g(b) \in U$, $b = f(a) \in V$. Then $[(w-b)^0]g = a$, and for $n > 0$,*

$$[(w-b)^n]g = \frac{1}{n!\,[z-a]f} \sum_{\sum_{j=2}^{\infty}(j-1)m_j = n-1} \left(\textstyle\sum_j j m_j\right)! \prod_{j=2}^{\infty} \frac{1}{m_j!}\left(-\frac{[(z-a)^j]f}{([z-a]f)^j}\right)^{m_j}.$$

Proof: Note that $f'(a) \neq 0$. Put $f_1(z) = f(a + z/f'(a)) - b$ and $g_1(w) = f'(a)(g(b+w) - a)$, so that $f_1(0) = 0 = g_1(0)$, $f_1 \circ g_1 = \mathrm{id}$ on a neighbourhood of 0, and $f_1'(0) = 1$. Write $f_1(z) = z(1 - h(z))$, where h is holomorphic in a neighbourhood of 0, and $h(0) = 0$. Then

$$\begin{aligned}
[w^n]g_1 = \frac{1}{n}[z^{-1}]\frac{1}{f_1^n} &= \frac{1}{n}[z^{n-1}]\frac{1}{(1-h)^n} \\
&= \frac{1}{n} \sum_{\sum_{j=1}^{\infty} jm_j = n-1} \frac{(\sum_j m_j)!}{m_1!\, m_2! \cdots} \frac{(\sum_j m_j + n - 1)!}{(\sum_j m_j)!\,(n-1)!} \prod_{j=1}^{\infty} \big([z^j]h\big)^{m_j} \\
&= \frac{1}{n!} \sum_{\sum_{j=1}^{\infty} jm_j = n-1} \frac{(\sum_j (j+1)m_j)!}{m_1!\, m_2! \cdots} \prod_{j=1}^{\infty} \big(-[z^{j+1}]f_1\big)^{m_j} \\
&= \frac{1}{n!} \sum_{\sum_{k=2}^{\infty} (k-1)m_k = n-1} \Big(\sum_k km_k\Big)! \prod_{k=2}^{\infty} \frac{\big(-[z^k]f_1\big)^{m_k}}{m_k!}
\end{aligned}$$

using Facts 3.1 and 3.2, and the expansion $[w^r](1-w)^{-n} = \binom{r+n-1}{n-1}$. The result follows by noting that for any $k, n > 0$, $[z^k]f_1 = ([(z-a)^k]f)/(f'(a))^k$, $[w^n]g_1 = f'(a)[(w-b)^n]g$, and $f'(a) = [z-a]f$. □

Let d be a constant. If f in Proposition 3.3 is a polynomial of degree d, then the product is nonzero only when $m_j = 0$ for every $j > d$, hence it suffices to enumerate $m_2, \dots, m_d$. It follows easily that the outer sum has polynomially many (namely, $O(n^d)$) terms, and we can compute $[(w-b)^n]g$ in uniform TC^0 given a, b, and the coefficients of f in binary, and n in unary.

Apart from a description of the coefficients, we also need bounds on the radius of convergence of the inverse series, and on its rate of convergence (i.e., on the norm of its coefficients). Generally speaking, the radius of convergence of a power series is the distance to the nearest singularity. Since a polynomial f is an entire proper map, its inverse cannot escape to infinity or hit a point where f is undefined, thus the only singularities that can happen are branch points. These occur at zeros of f'. This suggests that the main parameter governing the radius of convergence and other properties of the inverse should be the distance of a to the set $C_f = \{z \in \mathbb{C} : f'(z) = 0\}$ of *critical points* of f.

Lemma 3.4 *Let $f \in \mathbb{C}[x]$ be a degree d polynomial with no roots in $B(a, R)$, $R > 0$, and let $\mu > 0$. Then $|f(z) - f(a)| < \big((1+\mu)^d - 1\big)|f(a)|$ for all $z \in B(a, \mu R)$.*

Proof: Write $f(z) = c\prod_{j=1}^{d}(z - \alpha_j)$. We have

$$\frac{f(z)}{f(a)} = \prod_{j=1}^{d} \frac{(z-a) + (a-\alpha_j)}{a - \alpha_j} = \sum_{I \subseteq \{1,\dots,d\}} \prod_{j \in I} \frac{z-a}{a-\alpha_j},$$

hence

$$\left|\frac{f(z)}{f(a)} - 1\right| = \Big|\sum_{I \neq \varnothing} \prod_{j \in I} \frac{z-a}{a-\alpha_j}\Big| \leq \sum_{I \neq \varnothing} \prod_{j \in I} \frac{|z-a|}{R} < (1+\mu)^d - 1. \qquad \square$$

Proposition 3.5 *Let $f \in \mathbb{C}[x]$ have degree $d > 1$, $f(a) = b$, and $0 < R \leq \operatorname{dist}(a, C_f)$. Let*

$$g(w) = a + \sum_{n=1}^{\infty} c_n (w-b)^n$$

satisfy $f \circ g = \mathrm{id}_{B(b,\varrho)}$, where $\varrho > 0$ is the radius of convergence of g. Put

$$\mu = \sqrt[d-1]{2} - 1 \geq \frac{\ln 2}{d-1}, \qquad \nu = \frac{2(d-1)\mu - 1}{d} \geq \frac{\ln 4 - 1}{d},$$

$$\lambda_\delta = \sqrt[d]{1 + \delta d\nu} - 1 \geq \frac{\delta \ln\ln 4}{d}, \qquad \varrho_0 = \nu R\,|f'(a)|$$

for $0 < \delta \leq 1$ (the inequalities are established below). Then:

(i) f is injective on $\overline{B}(a, \mu R)$.

(ii) $\varrho \geq \varrho_0$.

(iii) $g[B(b,\varrho)] \supseteq B(a, \lambda_1 R)$, and more generally, $g[B(b, \delta\varrho_0)] \supseteq B(a, \lambda_\delta R)$ for each $\delta \in (0,1]$.

(iv) $|c_n| \leq \mu R / n \varrho_0^n$.

Proof: Notice that $e^x - 1 \geq x$ for every $x \in \mathbb{R}$, hence $\sqrt[d-1]{2} - 1 = \exp((\ln 2)/(d-1)) - 1 \geq (\ln 2)/(d-1)$; $\nu \geq (\ln 4 - 1)/d$ immediately follows. Similarly, $\lambda_\delta \geq \ln(1 + \delta(\ln 4 - 1))/d$. We have $\ln(1 + \delta(\ln 4 - 1)) \geq \delta \ln\ln 4$ for $\delta \in [0,1]$ as $\ln$ is concave.

(i): Let $u, v \in \overline{B}(a, \mu R)$, $u \neq v$. We have

$$f(v) - f(u) = \int_u^v f'(z)\,dz = (v-u)\left(f'(a) + \int_0^1 f'\big((1-t)u + tv\big) - f'(a)\,dt\right).$$

Since $|f'((1-t)u + tv) - f'(a)| < |f'(a)|$ for all $t \in (0,1)$ by Lemma 3.4, we obtain

$$\left|\int_0^1 f'\big((1-t)u + tv\big) - f'(a)\,dt\right| \leq \int_0^1 \left|f'\big((1-t)u + tv\big) - f'(a)\right|\,dt < |f'(a)|,$$

thus $f(u) \neq f(v)$.

(ii): Let $U = B(a, \mu R)$. Since f is a biholomorphism of U and $f[U]$, $\varrho \geq \operatorname{dist}(b, \mathbb{C} \smallsetminus f[U])$. Since $f[U]$ is open, there exists $w \notin f[U]$ such that $|w - b| = \operatorname{dist}(b, \mathbb{C} \smallsetminus f[U])$. Let $z_n \in U$ be such that $\lim_n f(z_n) = w$. By compactness, $\{z_n\}$ has a convergent subsequence; without loss of generality, there exists $z = \lim_n z_n$. Then $f(z) = w$ by continuity, hence $z \notin U$. However, $z \in \overline{U}$, hence z is in the topological boundary $\partial U = \overline{U} \smallsetminus \operatorname{int} U = \overline{U} \smallsetminus U$. We have thus verified that $\varrho \geq \operatorname{dist}(b, f[\partial U])$.

Let $u = a + \mu R e^{i\theta} \in \partial U$. We have

$$f(u) = b + \int_a^u f'(z)\,dz = b + Re^{i\theta}\left(\mu f'(a) + \int_0^\mu f'(a + te^{i\theta}R) - f'(a)\,dt\right).$$

By Lemma 3.4, $|f'(a + te^{i\theta}R) - f'(a)| \leq \big((1+t)^{d-1} - 1\big)|f'(a)|$, hence

$$\left|\int_0^\mu f'(a + te^{i\theta}R) - f'(a)\,dt\right| \leq |f'(a)| \int_0^\mu (1+t)^{d-1} - 1\,dt = |f'(a)|\left(\frac{(1+\mu)^d - 1}{d} - \mu\right)$$
$$= |f'(a)|\,\frac{2(1+\mu) - 1 - d\mu}{d} = |f'(a)|\,\frac{1 - (d-2)\mu}{d}.$$

Thus,

$$|f(u)-b| \geq R\,|f'(a)| \left(\mu - \frac{1-(d-2)\mu}{d}\right) = \nu R\,|f'(a)|.$$

(iii): The proof above shows that $g[B(b,\varrho_0)] \subseteq U$. As f is injective on $U \supseteq B(a,\lambda_\delta R)$, it suffices to show that $f[B(a,\lambda_\delta R)] \subseteq B(b,\delta\varrho_0)$. Let thus $u = a + \lambda R e^{i\theta}$, $\lambda < \lambda_\delta$. As above,

$$f(u) = b + Re^{i\theta}\left(\lambda f'(a) + \int_0^\lambda f'(a+te^{i\theta}R) - f'(a)\,dt\right)$$

and

$$\left|\int_0^\lambda f'(a+te^{i\theta}R) - f'(a)\,dt\right| \leq |f'(a)|\left(\frac{(1+\lambda)^d-1}{d} - \lambda\right),$$

hence

$$|f(u)-b| \leq R\,|f'(a)|\frac{(1+\lambda)^d-1}{d} < R\,|f'(a)|\frac{(1+\lambda_\delta)^d-1}{d} = \delta\varrho_0.$$

(iv): Let $\gamma(t) = a + \mu R e^{2\pi i t}$. By Fact 3.1 and Cauchy's integral formula,

$$c_n = \frac{1}{2\pi i n}\int_\gamma \frac{dz}{(f(z)-b)^n} = \frac{\mu R}{n}\int_0^1 \frac{e^{2\pi i t}\,dt}{(f(\gamma(t))-b)^n}.$$

The proof of (ii) shows $|f(\gamma(t))-b| \geq \varrho_0$, hence

$$|c_n| \leq \frac{\mu R}{n}\int_0^1 \frac{dt}{|f(\gamma(t))-b|^n} \leq \frac{\mu R}{n\varrho_0^n}.$$

□

Example 3.6 Let $f(z) = z^d$, $a = b = 1$. Then $f' = dz^{d-1}$, $C_f = \{0\}$, $R = 1$, $f'(a) = d$. It is not hard to see that f is injective on $B(1,r)$ iff no two points of $B(1,r)$ have arguments differing by $2\pi/d$ iff $r \leq \sin(\pi/d) = \pi/d + O(d^{-3})$. Since g must hit a root of f' at the circle of convergence, we must have $\varrho = 1 = (1/d)Rf'(a)$. Finally, $|(1+z)^d - 1|$ is maximized on $\{z : |z| = r\}$ for z positive real, thus $B(1,\lambda R) \subseteq g[B(1,\delta\varrho)]$ iff $(1+\lambda)^d - 1 \leq \delta$ iff $\lambda \leq (1+\delta)^{1/d} - 1 = \ln(1+\delta)/d + O(d^{-2})$. Thus, in Proposition 3.5, μ, ν, and λ_δ are optimal up to a linear factor.

Remark 3.7 We prefer to give a simple direct proof of Proposition 3.5 for the benefit of the reader. Nevertheless, we could have assembled the bounds (with somewhat different constants) from several more sophisticated results in the literature. The Grace–Heawood theorem (or rather its corollary, originally due to Alexander, Kakeya, and Szegő; see [21, Thm. 23,2]) states that (i) holds with $\mu = \sin(\pi/d)$ (which is tight in view of the z^d example). Then the Koebe 1/4-theorem [11, Thm. 14.7.8] implies (ii) with $\nu = \mu/4$, and one more application of the theorem yields (iii) with $\lambda_\delta = \nu\delta/4$.

4 Root finding in TC^0

We start with the core part of our root-finding algorithm. While it is conceptually simple, its output is rather crude, so we will have to combine it with some pre- and postprocessing to obtain the desired result (Theorem 4.5).

Theorem 4.1 *Let d be a constant. There exists a uniform TC^0 function which, given the coefficients of a degree d polynomial $f \in (\mathbb{Q}(i))[x]$ in binary and t in unary, computes a list $\{z_j : j < s\} \subseteq \mathbb{Q}(i)$ such that every complex root of f is within distance 2^{-t} of some z_j.*

Proof: If $d = 1$, it suffices to divide the coefficients of f. Assume $d \geq 2$. Let $\mu, \nu, \lambda = \lambda_{1/2}$ be as in Proposition 3.5 (more precisely, we should use their fixed rational approximations; we will ignore this for simplicity). Let $A = 1 + \lambda/5$, $p = \lceil 5\pi/\lambda \rceil$, and $\xi = e^{2\pi i/p}$ (approximately, again). Consider the TC^0 algorithm given by the following description:

(i) Input: $f = \sum_{j \leq d} f_j z^j$ with $f_j \in \mathbb{Q}(i)$, $f_d \neq 0$, and $t > 0$ in unary.

(ii) Put $\varepsilon = 2^{-t}$. Compute recursively a list $C = \{\alpha_j : j < s\}$ including $\varepsilon/4$-approximations of all roots of f'.

(iii) Output (in parallel) each α_j.

(iv) Put $c = 2 + \max_{j<d} |f_j/f_d|$ and $k_{\max} = \lceil \log(2c\varepsilon^{-1}) / \log A \rceil$.

(v) For every $j < s$, $k < k_{\max}$, and $q < p$, do the following in parallel.

(vi) Let $a = \alpha_j + \varepsilon A^k \xi^q$, $b = f(a)$, $R = \frac{1}{2}|a - \alpha_j|$, $N = \lceil \log_2(\mu R \varepsilon^{-1}) \rceil$.

(vii) For each $h \leq d$, let $\tilde{f}_h = \sum_{u=h}^{d} \binom{u}{h} f_u a^{u-h}$.

(viii) Compute and output

$$z_{j,k,q} = a + \sum_{\substack{m_2,\ldots,m_d \\ \sum_h (h-1)m_h < N}} \frac{(2m_2 + \cdots + dm_d)!\,(-\tilde{f}_2)^{m_2} \cdots (-\tilde{f}_d)^{m_d} (-b)^{1+m_2+\cdots+(d-1)m_d}}{m_2! \cdots m_d!\,(1 + m_2 + \cdots + (d-1)m_d)!\,\tilde{f}_1^{1+2m_2+\cdots+dm_d}}.$$

Let $f(\alpha) = 0$, we have to show that one of the numbers output by the algorithm is ε-close to α. If $|\alpha - \alpha_j| < \varepsilon$ for some j, we are done by step (iii). We can thus assume $\mathrm{dist}(\alpha, C) \geq \varepsilon$, which implies $\mathrm{dist}(\alpha, C_f) \geq 3\varepsilon/4$. Assume that α_j is an $\varepsilon/4$-approximation of the root $\tilde{\alpha}_j$ of f' nearest to α. Since all roots of f or f' have modulus bounded by $c - 1$ by Cauchy's bound, we have $\varepsilon \leq |\alpha - \alpha_j| < 2c$, thus there exists $k < k_{\max}$ such that $\varepsilon A^k \leq |\alpha - \alpha_j| < \varepsilon A^{k+1}$. Let $q < p$ be such that the argument of $\alpha - \alpha_j$ differs from $2\pi q/p$ by at most π/p, and consider steps (v)–(viii) for this particular choice of j, k, q (cf. Fig. 1). We have

$$|\alpha - a| \leq \left(\frac{\pi}{p} + 1 - \frac{1}{A} \right) |\alpha - \alpha_j| \leq \frac{2\lambda}{5} |\alpha - \alpha_j| < \frac{1}{5} |\alpha - \alpha_j|.$$

Notice that

$$\mathrm{dist}(\alpha, C_f) = |\alpha - \tilde{\alpha}_j| \geq |\alpha - \alpha_j| - \frac{\varepsilon}{4}$$

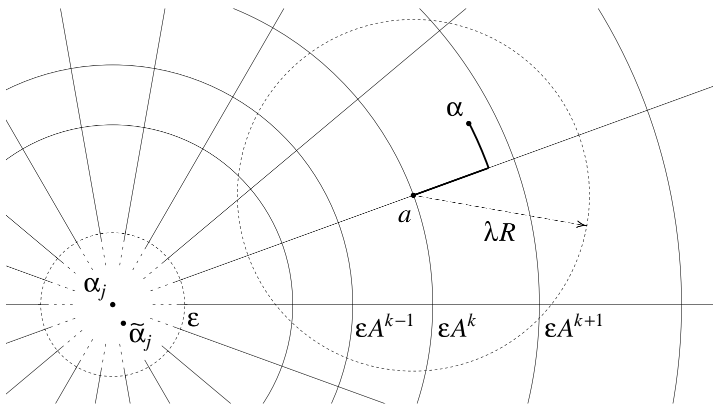

Figure 1: The spiderweb.

by the choice of $\tilde\alpha_j$ and α_j, hence

$$\operatorname{dist}(a, C_f) \geq \operatorname{dist}(\alpha, C_f) - |a-\alpha| \geq |\alpha-\alpha_j| - \frac{\varepsilon}{4} - \frac{1}{5}\,|\alpha-\alpha_j| \geq \frac{1}{2}\,|\alpha-\alpha_j| \geq R.$$

Since

$$|a-\alpha_j| \geq |\alpha-\alpha_j| - |a-\alpha| > \frac{4}{5}\,|\alpha-\alpha_j|,$$

we also have

$$|a-\alpha| < \frac{5}{4}\,\frac{2\lambda}{5}\,|a-\alpha_j| = \lambda R.$$

Let

$$g(w) = a + \sum_{n=1}^{\infty} c_n (w-b)^n$$

be an inverse of f in a neighbourhood of b, and let ϱ be its radius of convergence. By Proposition 3.5, $|-b| = |f(\alpha)-b| < \varrho_0/2$, where $\varrho_0 = \nu R\,|f'(a)| \leq \varrho$. Thus, $g(f(\alpha)) = g(0) = \alpha$. Since $\sum_h \tilde f_h z^h = f(z+a)$ by the binomial formula, $\tilde f_h = [(z-a)^h]f$. Then it follows from Proposition 3.3 that

$$z_{j,k,q} = a + \sum_{n=1}^{N} c_n (-b)^n.$$

Since

$$|c_n(-b)^n| \leq \frac{\mu R}{n\varrho_0^n}|b|^n < \frac{\mu R}{2^n}$$

by Proposition 3.5, we have

$$|\alpha - z_{j,k,q}| = \left|\sum_{n=N+1}^{\infty} c_n(-b)^n\right| < \frac{\mu R}{2^N} \leq \varepsilon. \qquad \square$$

Most of the algorithm described in Theorem 4.1 is independent of the assumption of d being constant (or it can be worked around). There are two principal exceptions. First, the recursion in step (ii) amounts to d sequential invocations of the algorithm. Second, while N is still linear in the size of the input, the main sum in step (viii) has roughly N^d terms. Thus, approximation of roots of arbitrary univariate polynomials can be done by (uniform) threshold circuits of depth $O(d)$ and size $n^{O(d)}$, where n is the total length of the input. (The known NC algorithms for root finding can do much better for large d.)

The algorithm from Theorem 4.1 does the hard work in locating the roots of f, but it suffers from several drawbacks:

- Its output includes a lot of bogus results that are not actually close to any root of f.
- There may be many elements on the list close to the same root, and we do not get any information on the multiplicity of the roots.
- The roots have no "identity": if we run the algorithm for two different ts, we do not know which approximate roots on the output lists correspond to each other.
- It may be desirable to output the binary expansions of the roots rather than just approximations.

We are going to polish the output of the algorithm to fix these problems. Let us first formulate precisely the goal.

Definition 4.2 The *t-digit binary expansion* of $a \in \mathbb{C}$ is the pair $\langle \lfloor \mathrm{Re}(a2^t) \rfloor, \lfloor \mathrm{Im}(a2^t) \rfloor \rangle$, where both integers are written in binary. A *root-finding algorithm* for a set of polynomials $P \subseteq (\mathbb{Q}(i))[x]$ is an algorithm with the following properties:

(i) The input consists of a polynomial $f \in P$ given by a list of its coefficients in binary, and a positive integer t in unary.

(ii) The output is a list of pairs $\{\langle z_j(f,t), e_j(f,t) \rangle : j < s(f,t)\}$.

(iii) For every $f \in P$, there exists a factorization

$$f(z) = c \prod_{j<s} (z - a_j)^{e_j},$$

where $c \in \mathbb{Q}(i)$, $a_j \in \mathbb{C}$, $a_j \neq a_k$ for $j \neq k$, and $e_j > 0$, such that for every t: $s(f,t) = s$, $e_j(f,t) = e_j$, and $z_j(f,t)$ is the t-digit binary expansion of a_j.

We note that the choice of base 2 in the output is arbitrary, the algorithm can output expansions in any other base if needed.

Lemma 4.3 *Let d be a constant. Given a degree d polynomial $f \in (\mathbb{Q}(i))[x]$, we can compute in uniform TC^0 a list of pairwise coprime square-free nonconstant polynomials f_j, $c \in \mathbb{Q}(i)$, and integers $e_j > 0$ such that $f = c \prod_{j<k} f_j^{e_j}$, where $k, e_j \leq d$.*

Proof: Since d is constant, division of degree d polynomials takes $O(1)$ arithmetical operations, hence it can be implemented in uniform TC^0. The same holds for gcd, using the Euclidean algorithm. We compute a list $L = \langle f_j : j < k \rangle$, $k \leq d$, of nonconstant polynomials such that $f = \prod_j f_j$ as follows:

(i) Start with $L = \langle f \rangle$. Repeat the following steps until none of them is applicable.

(ii) If f_j is not square-free, replace it with $\gcd(f_j, f_j')$ and $f_j / \gcd(f_j, f_j')$.

(iii) If $f_h \mid f_j$, $f_j \nmid f_h$ for some h, j, replace f_j in L with $f_h, f_j/f_h$.

(iv) If $g := \gcd(f_h, f_j) \neq 1$ for some h, j such that $f_h \nmid f_j$, $f_j \nmid f_h$, replace f_h, f_j in L with $g, g, f_h/g, f_j/g$.

The algorithm terminates after at most d steps, hence it is in TC^0. Clearly, it computes a list of square-free polynomials such that for every h, j, f_h is coprime to f_j or f_h is a scalar multiple of f_j. It remains to collect scalar multiples of the same polynomial together. □

Lemma 4.4 *Let d be a constant. Given a degree d square-free polynomial $f \in (\mathbb{Q}(i))[x]$ and t in unary, we can compute in uniform TC^0 a list $\{z_j : j < s\}$ such that every root of f is within distance 2^{-t} of some z_j, and every z_j is within distance 2^{-t} of some root.*

Proof: We use the notation from the proof of Theorem 4.1. We modify the algorithm from that proof as follows:

- We compute an $\varepsilon_0 > 0$ such that the distance of any root of f to any root of f' is at least ε_0 using Lemma 2.1. In step (ii), we put $\varepsilon = \min(2^{-t}, \varepsilon_0/3)$.
- We skip step (iii).
- In step (vi), we check that $|b| < \frac{1}{2}\nu|f'(a)|R$ and $|a - \alpha_{j'}| \geq R + \varepsilon/4$ for every $j' < s$. If either condition is violated, we output a symbol "$*$" instead of a number, and skip the remaining two steps.

The result is a list of numbers and $*$'s; it is easy to construct the sublist consisting of only numbers by a TC^0 function.

Let $z_{j,k,q}$ be one of the numbers output by the algorithm. In step (vi) we ensured $\operatorname{dist}(a, C) \geq R + \varepsilon/4$, hence $\operatorname{dist}(a, C_f) \geq R$. Moreover, $|0 - b| < \varrho_0/2$, hence 0 is within the radius of convergence of g, and $\alpha = g(0)$ is a root of f whose distance from $z_{j,k,q}$ is

$$|\alpha - z_{j,k,q}| = \left| \sum_{n=N+1}^{\infty} c_n(-b)^n \right| < \frac{\mu R}{2^N} \leq \varepsilon.$$

On the other hand, let α be a root of f. Since $\operatorname{dist}(\alpha, C_f) \geq \varepsilon_0$, we have $\operatorname{dist}(\alpha, C) \geq \varepsilon$, hence we can choose j, k, q such that $|\alpha - z_{j,k,q}| < \varepsilon$ as in the proof of Theorem 4.1. We have to show that the extra conditions in step (vi) are satisfied. $|b| < \frac{1}{2}\nu R|f'(a)|$ was verified in the proof of Theorem 4.1. Moreover,

$$|a - \alpha_{j'}| \geq |\alpha - \tilde{\alpha}_{j'}| - |a - \alpha| - \frac{\varepsilon}{4} \geq |a - \tilde{\alpha}_j| - |a - \alpha| - \frac{\varepsilon}{4} \geq \frac{4}{5}|\alpha - \alpha_j| - \frac{\varepsilon}{2} \geq R + \frac{\varepsilon}{4}$$

as $|\alpha - \alpha_j| \geq \varepsilon_0 - \varepsilon/4 > \frac{5}{2}\varepsilon$. □

We can now finish the proof of the main result of this paper:

Theorem 4.5 *For every constant d, there exists a uniform* TC^0 *root-finding algorithm for degree d polynomials in the sense of Definition* 4.2.

Proof: We employ the notation of Definition 4.2. By Lemma 4.3, we can assume f to be square-free (in which case we will have $e_j(f,t) = 1$ for all j, so we only need to compute the roots). Consider the following TC^0 algorithm:

(i) Using Lemma 2.1, compute an $\eta > 0$ such that all roots of f are at distance at least η from each other.

(ii) Using Lemma 4.4, compute a list $\{r'_j : j < u\}$ such that every root of f is within distance $\eta/5$ of some r'_j, and vice versa.

(iii) Note that if r'_h and r'_j correspond to the same root, then $|r'_h - r'_j| < \frac{2}{5}\eta$, otherwise $|r'_h - r'_j| > \frac{3}{5}\eta$. Use this criterion to omit duplicate roots from the list, creating a list $\{r_j : j < d\}$ which contains $\eta/5$-approximations of all roots of f, each of them exactly once.

(iv) If $\varepsilon := 2^{-t} \geq \eta/5$, output $z_j := r_j$ and halt. Otherwise use Lemma 4.4 to construct a list $\{z'_h : h < s\}$ consisting of ε-approximations of roots of f.

(v) For each $j < d$, output $z_j := z'_{h(j)}$, where $h(j)$ is the smallest $h < s$ such that $|z'_h - r_j| < \eta/2$.

Notice that the computation of r_j is independent of t. Let a_j be the unique root of f such that $|a_j - r_j| < \eta/5$. Given t and i, let j' be such that $|z'_h - a_{j'}| < \varepsilon$. Then $|z'_h - r_j| < \varepsilon + \eta/5 \leq \frac{2}{5}\eta$ if $j = j'$, otherwise $|z'_h - r_j| > \frac{4}{5}\eta - \varepsilon \geq \frac{3}{5}\eta$. Thus, the definition of $h(j)$ in the last step is sound, and guarantees $|z_j - a_j| < \varepsilon$.

It follows that this TC^0 function has all the required properties, except that it computes approximations instead of binary expansions. We can fix this as follows. Using the algorithm we have just described, we can compute integers u, v such that $|u + iv - 2^t a_j| < 1$. Then $\lfloor \mathrm{Re}(2^t a_j) \rfloor$ is either u or $u - 1$, hence it remains to find the sign of $\mathrm{Re}(2^t a_j) - u$ (the case of Im is similar).

Let $g(z) = f(2^{-t}(2z + u))$, $h(z) = \overline{g}(-z)$, and $\alpha = \frac{1}{2}(2^t a_j - u)$. Then $g(\alpha) = 0 = h(-\overline{\alpha})$ and $\alpha - (-\overline{\alpha}) = \mathrm{Re}(2^t a_j) - u$. Using Lemma 2.1, we can compute $\xi > 0$ such that $|\alpha - (-\overline{\alpha})| \geq \xi$ whenever it is nonzero. Using the algorithm above, we can compute rational u', v' such that $|u' + iv' - 2^t a_j| < \xi/4$. If $|u - u'| < \xi/2$, then $\mathrm{Re}(2^t a_j) = u$. Otherwise, $|\mathrm{Re}(2^t a_j) - u| \geq \xi$, hence the sign of $u' - u$ agrees with the sign of $\mathrm{Re}(2^t a_j) - u$. □

Corollary 4.6 *If α is a fixed real algebraic number, then the kth bit of α can be computed in uniform* TC^0*, given k in unary.* □

(Note that this corollary is only interesting in the uniform setting, since the language is unary.)

5 Open induction in VTC^0

As we already mentioned in the introduction, our primary motivation for studying root finding for constant-degree polynomials comes from bounded arithmetic. We will now describe the connection in more detail. A reader not interested in bounded arithmetic may safely stop reading here.

The basic objects of study in bounded arithmetic are weak first-order theories based on integer arithmetic. There is a loose correspondence of arithmetical theories to complexity classes: in particular, if a theory T corresponds to a class C, then the provable total computable functions of T are functions from C (or more precisely, FC). The following is one of the natural problems to study in this context: assume we have a concept (say, a language or a function) from the computational class C. Which properties of this concept are provable in the theory T? (This asks for a form of feasible reasoning: what can we show about the concept when we are restricted to tools not exceeding its complexity?)

Here we are concerned with the theory VTC^0, corresponding to TC^0. We refer the reader to Cook and Nguyen [12] for a comprehensive treatment of VTC^0. Let us briefly recall that VTC^0 is a two-sorted theory, with one sort intended for natural numbers (which we think of as given in unary), and one sort for finite sets of these unary numbers (which we also regard as finite binary strings, or as numbers written in binary). We are primarily interested in the binary number sort, we consider the unary sort to be auxiliary. We use capital letters $X, Y, \dots$ for variables of the binary (set) sort, and lowercase letters $x, y, \dots$ for the unary sort. The language of the theory consists of basic arithmetical operations on the unary sort, the elementhood (or bit) predicate $x \in X$, and a function $|X|$ which extracts an upper bound on elements of a set X. The axioms of VTC^0 include comprehension for Σ^B_0 formulas (formulas with number quantifiers bounded by a term and no set quantifiers)—which also implies induction on unary numbers for Σ^B_0 formulas—and an axiom ensuring the existence of counting functions for any set. The provably total computable (i.e., Σ^1_1-definable: Σ^1_1 formulas consist of a block of existential set quantifiers in front of a Σ^B_0 formula) functions of VTC^0 are the TC^0 functions.

In VTC^0, we can define the basic arithmetical operations $+, \cdot, \leq$ on binary integers. Our main question is, what properties of these operations are provable in VTC^0. (We can make this more precise as follows: which theories in the usual single-sorted language of arithmetic $L_{PA} = \langle 0, 1, +, \cdot, \leq \rangle$ are interpreted in VTC^0 by the corresponding operations on the binary sort?) It is not hard to show that VTC^0 proves binary integers to form a discretely ordered ring (DOR). What we would especially like to know is whether VTC^0 can prove the induction schema on the binary sort

$$\varphi(0) \wedge \forall X \, (\varphi(X) \to \varphi(X+1)) \to \forall X \, \varphi(X)$$

for some nontrivial class of formulas φ. In particular, we want to know whether VTC^0 includes the theory *IOpen* (axiomatized by induction for open formulas of L_{PA} over DOR) introduced by Shepherdson [32] and widely studied in the literature.

Now, assume for a moment that $VTC^0 \vdash IOpen$. Then for each constant d, VTC^0 proves

$$X < Y \wedge F(X) \leq 0 < F(Y) \rightarrow \exists Z\,(X \leq Z < Y \wedge F(Z) \leq 0 < F(Z+1))$$

where $F(X) = \sum_{j \leq d} U_j X^j$ is a degree d integer polynomial whose coefficients are parameters of the formula. This is (equivalent to) a Σ^1_1 formula, hence the existential quantifier is, provably in VTC^0, witnessed by a TC^0 function $G(U_0, \dots, U_d, X, Y)$. Since any rational polynomial is a scalar multiple of an integer polynomial, and we can pass from a polynomial $F(X)$ to $2^{td}F(2^{-t}X)$ to reduce the error from 1 to 2^{-t}, we see that there is a TC^0 algorithm solving the following root-finding problem: given a degree d rational polynomial and two rational bounds where it assumes opposite signs, approximate a real root of the polynomial between the two bounds up to a given accuracy. Using a slightly more complicated argument, one can also obtain a root-finding algorithm in the set-up we considered earlier : i.e., we approximate all complex roots of the polynomial, and the input of the algorithm is only the polynomial and the desired error of approximation. Thus, a TC^0 root-finding algorithm is a necessary prerequisite for showing *IOpen* in VTC^0.

We can in a sense reverse the argument above to obtain a proof of open induction from a root-finding algorithm, but there is an important caveat. The way we used the witnessing theorem for VTC^0, we lost the information that the soundness of the algorithm is provable in VTC^0. Indeed, if we are only concerned with the computational complexity of witnessing functions, then witnessing of Σ^1_1 formulas is unaffected by addition of true universal (i.e., $\forall\Sigma^B_0$) axioms to the theory. In other words, the same argument shows the existence of a root-finding algorithm from the weaker assumption $VTC^0 + \mathrm{Th}_{\forall\Sigma^B_0}(\mathbb{N}) \vdash IOpen$, where $\mathrm{Th}_{\forall\Sigma^B_0}(\mathbb{N})$ denotes the set of all $\forall\Sigma^B_0$ sentences true in the standard model of arithmetic. Now, this formulation of the argument can be reversed:

Theorem 5.1 *The theory $VTC^0 + \mathrm{Th}_{\forall\Sigma^B_0}(\mathbb{N})$ proves IOpen for the binary number sort.*

Proof: Let M be a model of $VTC^0 + \mathrm{Th}_{\forall\Sigma^B_0}(\mathbb{N})$, and D be the discretely ordered ring of the binary integers of M. For any constant d, we can use Theorem 4.1 to construct a TC^0 function which, given the coefficients of an integer polynomial of degree d, computes a list of integers $a_0 < a_1 < \cdots < a_k$, $k \leq d$, such that the sign of the polynomial is constant on each of the *integer* intervals (a_j, a_{j+1}), $(-\infty, a_0)$, $(a_k, +\infty)$. This property of the function is expressible by a $\forall\Sigma^B_0$ sentence (when the coefficients of the polynomial and the a_j are taken from the binary sort), hence it holds in D that such elements $a_0, \dots, a_k$ exist for every polynomial over D.

Any atomic formula $\varphi(x)$ of L_{PA} with parameters from D is equivalent in DOR to the formula $f(x) \leq 0$ for some $f \in D[x]$, hence $\varphi(D) := \{x : D \models \varphi(x)\}$ is a finite union of intervals. Sets of this kind form a Boolean algebra, hence $\varphi(D)$ is a finite union of intervals for every open formula φ. This implies induction for φ: if $D \models \varphi(0) \wedge \neg\varphi(u)$ for some $u > 0$, the interval I of $\varphi(D)$ containing 0 cannot be infinite from above, hence its larger end-point $v \in D$ satisfies $D \models \varphi(v) \wedge \neg\varphi(v+1)$. $\square$

Problem 5.2 *Does VTC^0 prove IOpen?*

In light of the discussion above, Problem 5.2 is essentially equivalent to the following: are there TC^0 root-finding algorithms for constant-degree polynomials *whose correctness is provable in* VTC^0? We remark that the complex-analytic tools we used in the proof of Theorem 4.1 are not available in VTC^0.

We note that already proving the totality of integer division in VTC^0 (i.e., formalization of a TC^0 integer division algorithm in VTC^0) is a nontrivial open[1] problem, thus Problem 5.2 may turn out to be too ambitious a goal. The following is a still interesting version of the question, which may be easier to settle:

Problem 5.3 *Does $VTC^0 + IMUL$ prove IOpen, where IMUL is a natural axiom postulating the totality of iterated integer multiplication?*

We also mention that it is not hard to prove in VTC^0 that binary integers form a $\mathbb{Z}$-ring, which implies all universal consequences of *IOpen* in the language of ordered rings. The problem is thus only with statements with a genuinely existential import (note that *IOpen* is a $\forall\exists$ theory).

Acknowledgements

I am grateful to Paul Beame and Yuval Filmus for useful discussions, and to anonymous referees for helpful suggestions.

References

[1] Lars V. Ahlfors, *Complex analysis: An introduction to the theory of analytic functions of one complex variable*, McGraw–Hill, New York, 1979.

[2] Eric Allender, *The permanent requires large uniform threshold circuits*, Chicago Journal of Theoretical Computer Science 1999, article no. 7.

[3] ———, *Arithmetic circuits and counting complexity classes*, in: Complexity of computations and proofs (Caserta) (J. Krajíček, ed.), Quaderni di Matematica vol. 13, Seconda Universita di Napoli, 2004, pp. 33–72.

[4] David A. Mix Barrington, Neil Immerman, and Howard Straubing, *On uniformity within NC^1*, Journal of Computer and System Sciences 41 (1990), no. 3, pp. 274–306.

[5] Paul W. Beame, Stephen A. Cook, and H. James Hoover, *Log depth circuits for division and related problems*, SIAM Journal on Computing 15 (1986), no. 4, pp. 994–1003.

[1] Hesse et al. [16, Cor. 6.6] claim that the totality of integer division is provable in VTC^0 (or rather, in the theory C^0_2 of Johannsen and Pollett [18], RSUV-isomorphic to $VTC^0 + \Sigma^B_0\text{-}AC$, which is $\forall\Sigma^1_1$-conservative over VTC^0). However, the way it is stated there with no proof as an "immediate" corollary strongly suggests that the claim is due to a misunderstanding. See also [12, §IX.7.3].

[6] Michael Ben-Or, Ephraim Feig, Dexter Kozen, and Prasoon Tiwari, *A fast parallel algorithm for determining all roots of a polynomial with real roots*, SIAM Journal on Computing 17 (1988), no. 6, pp. 1081–1092.

[7] Ashok K. Chandra, Larry Stockmeyer, and Uzi Vishkin, *Constant depth reducibility*, SIAM Journal on Computing 13 (1984), no. 2, pp. 423–439.

[8] Andrew Y. Chiu, George I. Davida, and Bruce E. Litow, *Division in logspace-uniform* NC^1, RAIRO – Theoretical Informatics and Applications 35 (2001), no. 3, pp. 259–275.

[9] Louis Comtet, *Advanced combinatorics: The art of finite and infinite expansions*, D. Reidel Publishing Company, Dordrecht, 1974.

[10] John B. Conway, *Functions of one complex variable*, Springer, New York, 1978.

[11] ———, *Functions of one complex variable II*, Springer, New York, 1995.

[12] Stephen A. Cook and Phuong Nguyen, *Logical foundations of proof complexity*, Cambridge University Press, New York, 2010.

[13] Émile Durand, *Solutions numériques des équations algébriques. Tome I: Équations du type* $F(x) = 0$*: Racines d'un polynôme*, Masson, Paris, 1960 (in French).

[14] Gene H. Golub and Charles F. Van Loan, *Matrix computations*, third ed., Johns Hopkins University Press, Baltimore, 1996.

[15] András Hajnal, Wolfgang Maass, Pavel Pudlák, Márió Szegedy, and György Turán, *Threshold circuits of bounded depth*, Journal of Computer and System Sciences 46 (1993), no. 2, pp. 129–154.

[16] William Hesse, Eric Allender, and David A. Mix Barrington, *Uniform constant-depth threshold circuits for division and iterated multiplication*, Journal of Computer and System Sciences 65 (2002), no. 4, pp. 695–716.

[17] Michael A. Jenkins and Joseph F. Traub, *A three-stage variable-shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration*, Numerische Mathematik 14 (1970), no. 3, pp. 252–263.

[18] Jan Johannsen and Chris Pollett, *On proofs about threshold circuits and counting hierarchies* (*extended abstract*), in: Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science, 1998, pp. 444–452.

[19] Immo O. Kerner, *Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen*, Numerische Mathematik 8 (1966), no. 3, pp. 290–294 (in German).

[20] Alexis Maciel and Denis Thérien, *Efficient threshold circuits for power series*, Information and Computation 152 (1999), no. 1, pp. 62–73.

[21] Morris Marden, *The geometry of the zeros of a polynomial in a complex variable*, Mathematical Surveys vol. 3, American Mathematical Society, New York, 1949.

[22] C. Andrew Neff, *Specified precision polynomial root isolation is in NC*, Journal of Computer and System Sciences 48 (1994), no. 3, pp. 429–463.

[23] C. Andrew Neff and John H. Reif, *An efficient algorithm for the complex roots problem*, Journal of Complexity 12 (1996), no. 2, pp. 81–115.

[24] Victor Y. Pan, *Fast and efficient algorithms for sequential and parallel evaluation of polynomial zeros and of matrix polynomials*, in: Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science, 1985, pp. 522–531.

[25] ———, *Optimal and nearly optimal algorithms for approximating polynomial zeros*, Computers & Mathematics with Applications 31 (1996), no. 12, pp. 97–138.

[26] ———, *Solving a polynomial equation: Some history and recent progress*, SIAM Review 39 (1997), no. 2, pp. 187–220.

[27] Ian Parberry and Georg Schnitger, *Parallel computation with threshold functions*, Journal of Computer and System Sciences 36 (1988), no. 3, pp. 278–302.

[28] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical recipes: The art of scientific computing*, third ed., Cambridge University Press, 2007.

[29] John H. Reif, *Logarithmic depth circuits for algebraic functions*, SIAM Journal on Computing 15 (1986), no. 1, pp. 231–242.

[30] John H. Reif and Stephen R. Tate, *On threshold circuits and polynomial computation*, SIAM Journal on Computing 21 (1992), no. 5, pp. 896–908.

[31] Walter L. Ruzzo, *On uniform circuit complexity*, Journal of Computer and System Sciences 22 (1981), no. 3, pp. 365–383.

[32] John C. Shepherdson, *A nonstandard model for a free variable fragment of number theory*, Bulletin de l'Académie Polonaise des Sciences 12 (1964), no. 2, pp. 79–86.

[33] Christopher S. Wallace, *A suggestion for a fast multiplier*, IEEE Transactions on Electronic Computers 13 (1964), no. 1, pp. 14–17.

[34] Chee Keng Yap, *Fundamental problems in algorithmic algebra*, Oxford University Press, 2000.