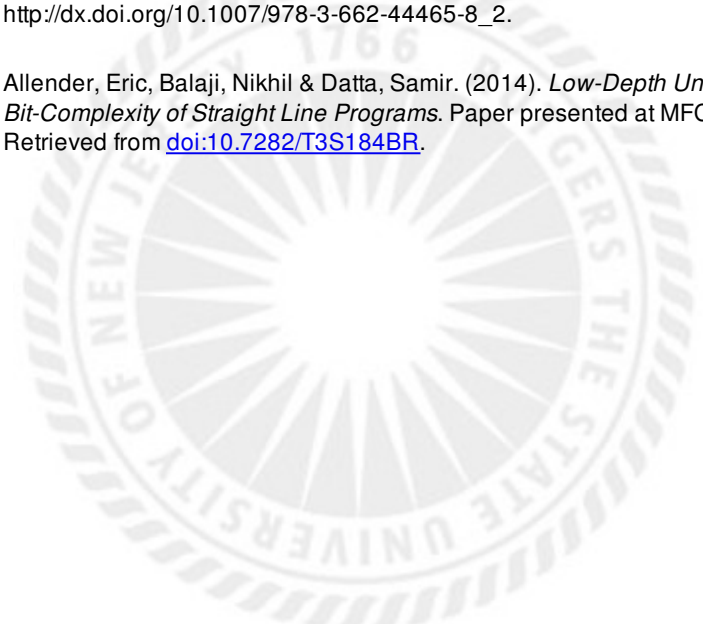
Low-Depth Uniform Threshold Circuits and the Bit-Complexity of Straight Line Programs



Low-Depth Uniform Threshold Circuits and the Bit-Complexity of Straight Line Programs

Eric Allender[1], Nikhil Balaji[2], and Samir Datta[2]

[1] Department of Computer Science, Rutgers University, USA
allender@cs.rutgers.edu
[2] Chennai Mathematical Institute, India
{nikhil,sdatta}@cmi.ac.in

Abstract. We present improved uniform TC^0 circuits for division, matrix powering, and related problems, where the improvement is in terms of "majority depth" (as studied by Maciel and Thérien). As a corollary, we obtain improved bounds on the complexity of certain problems involving arithmetic circuits, which are known to lie in the counting hierarchy.

1 Introduction

How hard is it to compute the 10^{100}-th bit of the binary expansion of $\sqrt{2}$? Datta and Pratap [6], and Jeřábek [15] considered the question of computing the m-th bit of an algebraic number. Jeřábek [15] showed that this problem has uniform TC^0 circuits[3] of size polynomial in m (which is not so useful when $m = 10^{100}$). Earlier, Datta and Pratap showed a related result: when m is expressed in *binary*, this problem lies in the counting hierarchy. More precisely, Datta and Pratap showed that this problem is reducible to the problem of computing certain bits of the quotient of two numbers represented by arithmetic circuits of polynomial size.[4] Thus, we are led to the problem of evaluating arithmetic circuits. **In this paper, we focus on arithmetic circuits *without input variables*.** Thus an arithmetic circuit is a (possibly very compact) representation of a number.

Arithmetic circuits of polynomial size can produce numbers that require exponentially-many bits to represent in binary. The problem[5] known as BitSLP ($= \{(C, i, b) :$ the i-th bit of the number represented by arithmetic circuit C is $b\}$) is known to be hard for $\#\mathsf{P}$ [3]. It was known that BitSLP lies in the counting hierarchy [3], but the best previously-known bound for this problem is the bound mentioned in [3] and credited there to [2]: $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}}$. That bound follows via a straightforward translation of a uniform TC^0 algorithm presented in [12].

[3] For somewhat-related TC^0 algorithms on sums of radicals, see [14].

[4] It is mistakenly claimed in [6] that this problem lies in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$. In this paper, we prove the weaker bound that it lies in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$.

[5] "SLP" stands for "straight-line program": a model equivalent to arithmetic circuits. Throughout the rest of the paper, we will stick with the arithmetic circuit formalism.

In this paper, we improve this bound on the complexity of BitSLP to $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$. In order to do this, we present improved uniform TC^0 algorithms for a number of problems that were already known to reside in uniform TC^0. The improvements that we provide are related to the *depth* of the TC^0 circuits. There are several possible variants of "depth" that one could choose to study. For instance, several papers have studied circuits consisting only of majority gates, and tight bounds are known for the depth required for several problems, in that model. (See, for instance [10,24,27,23] and other work referenced there.) Since our motivation comes largely from the desire to understand the complexity of problems in the counting hierarchy, it turns out that it is much more relevant to consider the notion of *majority depth* that was considered by Maciel and Thérien [20]. In this model, circuits have unbounded-fan-in AND, OR, and MAJORITY gates (as well as NOT gates). The class $\widehat{\mathsf{TC}}^0_d$ consists of functions computable by families of threshold circuits of polynomial size and constant depth such that no path from an input to an output gate encounters more than d MAJORITY gates. Thus the class of functions with majority depth zero, $\widehat{\mathsf{TC}}^0_0$, is precisely AC^0. In order to explain the connection between $\widehat{\mathsf{TC}}^0_d$ and the counting hierarchy, it is necessary to define the levels of the counting hierarchy.

Define $\mathsf{CH}_1 = \mathsf{PP}$, and $\mathsf{CH}_{k+1} = \mathsf{PP}^{\mathsf{CH}_k}$.

Proposition 1. *(Implicit in [3, Theorem 4.1].) Let A be a set such that for some k, some poly-time function f and some dlogtime-uniform $\widehat{\mathsf{TC}}^0_d$ circuit family C_n: $x \in A$ iff $C_{|x|+2^{|x|^k}}(x, f(x,1)f(x,2)\ldots f(x,2^{|x|^k}))$ accepts. Then $A \in \mathsf{PH}^{\mathsf{CH}_d}$.*

(One important part of the proof of Proposition 1 is the fact that, by Toda's theorem [25], for every oracle A, $\mathsf{PP}^{\mathsf{PH}^A} \subseteq \mathsf{P}^{\mathsf{PP}^A}$. Thus all of the AC^0 circuitry inside the $\widehat{\mathsf{TC}}^0_d$ circuit can be swallowed up by the PH part of the simulation.)

Note that the dlogtime-uniformity condition is crucial for Proposition 1. Thus, for the remainder of this paper, all references to $\widehat{\mathsf{TC}}^0_d$ will refer to the dlogtime-uniform version of this class, unless we specifically refer to nonuniform circuits. Table 1 compares the complexity bounds that Maciel and Thérien obtained in the *nonuniform* setting with the bounds that we are able to obtain in the uniform setting. (Maciel and Thérien also considered several problems for which they gave uniform circuit bounds; the problems listed in Table 1 were not known to lie in dlogtime-uniform TC^0 until the subsequent work of [12].) All previously-known dlogtime-uniform TC^0 algorithms for these problems rely on the CRR-to-binary algorithm of [12], and thus have at *least* majority-depth 4 (as analyzed by [2]); no other depth analysis beyond $O(1)$ was attempted.

In all of the cases where our uniform majority-depth bounds are worse than the nonuniform bounds given by [20], our algorithms also give rise to nonuniform algorithms that match the bounds of [20] (by hardwiring in some information that depends only on the length), although in all cases the algorithms differ in several respects from those of [20].

Problem	Nonuniform Majority-Depth [20]	Uniform Majority-Depth [This Paper]
Iterated multiplication	3	3
Division	2	3
Powering	2	3
CRR-to-binary	1	3
Matrix powering	O(1) [21,12]	3

All of the TC^0 algorithms that are known for the problems considered in this paper rely on partial evaluations or approximations. The technical innovations in our improved algorithms rely on introducing yet another approximation, as discussed in Lemmas 2 and 3.

Table 1 also lists one problem that was not considered by Maciel and Thérien: given as input 1^m and a $k \times k$ matrix A, produce A^m. For any fixed k, this problem was put in nonuniform TC^0 [21]; and by [12] it is also in dlogtime-uniform TC^0. The corresponding problem of computing *large* powers of a $k \times k$ matrix (i.e., when m is given in *binary*) has been discussed recently; see the final section of [22]. We show that this version of matrix powering is in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$.

In addition to BitSLP, there has also been interest in the related problem PosSLP ($= \{C :$ the number represented by arithmetic circuit C is positive$\}$) [9,18,17,19]. $\mathsf{PosSLP} \in \mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$, and is not known to be in PH [3], but in contrast to BitSLP, it is not known (or believed [9]) to be NP-hard. Our theorems do not imply any new bounds on the complexity of PosSLP, but we do conjecture that BitSLP and PosSLP both lie in $\mathsf{PH}^{\mathsf{PP}}$. This conjecture is based mainly on the heuristic that says that, for problems of interest, if a nonuniform circuit is known, then corresponding dlogtime-uniform circuits usually also exist. Converting from CRR to binary can be done nonuniformly in majority-depth one, and there is no reason to believe that this is not possible uniformly – although it seems clear that a different approach will be needed, to reach this goal.

The well-studied Sum-of-Square-Roots problem reduces to PosSLP [3], which in turn reduces to BitSLP. But the relationship between PosSLP and the matrix powering problem (given a matrix A and n-bit integer j, output the j^{th} bit of a given entry of A^j) is unclear, since matrix powering corresponds to evaluating *very restricted* arithmetic circuits. Note that some types of arithmetic involving large numbers *can* be done in P; see [13]. Might matrix powering also lie in PH?

We provide a very weak "hardness" result for the problem of computing the bits of large powers of 2-by-2 matrices, to shed some dim light on this question. We show that the Sum-of-Square-Roots problem reduces to this problem via $\mathsf{PH}^{\mathsf{PP}}$-Turing reductions. Due to lack of space, we defer the proof to the full version of this paper.

2 Preliminaries

Given a list of primes $\Pi = (p_1, \ldots, p_m)$ and a number X, the CRR_Π representation of X is the list $(X \bmod p_1, \ldots, X \bmod p_m)$. We omit the subscript Π if context makes clear. For more on complexity classes such as $\mathsf{AC}^0, \mathsf{TC}^0, \mathsf{NC}^1$, as well as a discussion of dlogtime uniformity, see [26].

3 Uniform Circuits for Division

Theorem 1. *The function taking as input $X \in [0, 2^n), Y \in [1, 2^n)$, and 0^m and producing as output the binary expansion of X/Y correct to m places is in $\widehat{\mathsf{TC}}^0_3$.*

Proof. This task is trivial if $Y = 1$; thus assume that $Y \geq 2$. Computing the binary expansion of Z/Y correct to m places is equivalent to computing $\lfloor 2^m Z/Y \rfloor$. Thus we will focus on the task of computing $\lfloor X/Y \rfloor$, given integers X and Y.

Our approach will be to compute $\widetilde{V}(X, Y)$, a strict underestimate of X/Y, such that $X/Y - \widetilde{V}(X, Y) < 1/Y$. Since $Y > 1$, we have that $\lfloor X/Y \rfloor \neq \lfloor (X + 1)/Y \rfloor$ if and only if $(X+1)/Y = \lfloor X/Y \rfloor + 1$. It follows that in *all* cases $\lfloor X/Y \rfloor = \lfloor \widetilde{V}(X + 1, Y) \rfloor$, since

$$\left\lfloor \frac{X}{Y} \right\rfloor \leq \frac{X}{Y} = \frac{X+1}{Y} - \frac{1}{Y} < \widetilde{V}(X+1, Y) < \frac{X+1}{Y}.$$

Note: in order to compute $\lfloor \frac{X}{Y} \rfloor$, we compute an approximation to $(X + 1)/Y$.

The approximation $\widetilde{V}(X, Y)$ is actually defined in terms of another rational approximation $W(X, Y)$, which will have the property that $\widetilde{V}(X, Y) \leq W(X, Y) < X/Y$. We postpone the definition of $\widetilde{V}(X, Y)$, and focus for now on $W(X, Y)$, an under approximation of $\frac{X}{Y}$ with error at most $2^{-(n+1)}$. .

Using AC^0 circuitry, we can compute a value t such that $2^{t-1} \leq Y < 2^t$.

Let $u = 1 - 2^{-t}Y$. Then $u \in (0, \frac{1}{2}]$. Thus, $Y^{-1} = 2^{-t}(1-u)^{-1} = 2^{-t}(1 + u + u^2 + \ldots)$. Set $Y' = 2^{-t}(1 + u + u^2 + \ldots + u^{2n+1})$, then

$$0 < Y^{-1} - Y' \leq 2^{-t} \sum_{j > 2n+1} 2^{-j} < 2^{-(2n+1)}$$

Define $W(X, Y)$ to be XY'. Hence, $0 < \frac{X}{Y} - W(X, Y) < 2^{-(n+1)}$.

We find it useful to use this equivalent expression for $W(X, Y)$:

$$W(X, Y) = \frac{X}{2^t} \sum_{j=0}^{2n+1} (1 - \frac{Y}{2^t})^j = \frac{1}{2^{2(n+1)t}} \sum_{j=0}^{2n+1} X(2^t - Y)^j 2^{(2n+1-j)t};$$

$W(X, Y) = \frac{1}{2^{2(n+1)t}} \sum_{j=0}^{2n+1} W_j(X, Y)$, where $W_j(X, Y) = X(2^t - Y)^j (2^{(2n+1-j)t})$.

Lemma 1. *(Adapted from [6]) Let Π be any set of primes such that the product M of these primes lies in $(2^{n^c}, 2^{n^d})$ for some $d > c \geq 3$. Then, given X, Y, Π we can compute the CRR_Π representations of the $2(n+1)$ numbers $W_j(X, Y)$ (for $j \in \{0, \ldots, 2n+1\}$) in $\widehat{\mathsf{TC}}^0_1$.*

Proof. Using AC^0 circuitry, we can compute $2^t - Y$, $2^j \bmod p$ for each prime $p \in \Pi$ and various powers j, as well as finding generators mod p. In $\widehat{\mathsf{TC}}^0_1$ we can compute $X \bmod p$ and $(2^t - Y) \bmod p$ (each of which has $O(\log n)$ bits). Using those results, with AC^0 circuitry we can compute the powers $(2^t - Y)^j \bmod p$ and then do additional arithmetic on numbers of $O(\log n)$ bits to obtain the product $X(2^t - Y)^j(2^{(2n+1-j)t}) \bmod p$ for each $p \in \Pi$. (The condition that $c \geq 3$ ensures that the numbers that we are representing are all less than M.) □

Having the CRR_Π representation of the number $W_j(X,Y)$, our goal might be to convert the $W_j(X,Y)$ to binary, and take their sum. For efficiency, instead we compute an approximation (in binary) to $W(X,Y)/M$ where $M = \prod_{p\in\Pi} p$. In Lemma 3 we build on this to compute our approximation $\widetilde{V}(X,Y)$ to $W(X,Y)$.

Recall that $W(X,Y) = \frac{1}{2^{2(n+1)t}} \sum_{j=0}^{2n+1} W_j(X,Y)$. Thus $2^{2(n+1)t} W(X,Y)$ is an integer with the same significant bits as $W(X,Y)$.

Lemma 2. *Let Π be any set of primes such that the product M of these primes lies in $(2^{n^c}, 2^{n^d})$ for a fixed constant $d > c \geq 3$, and let b be any natural number. Then, given X, Y, Π we can compute the binary representation of a* good *approximation to $\frac{2^{2(n+1)t}W(X,Y)}{M}$ in $\widehat{\mathsf{TC}}^0_2$ (where by good we mean that it under-estimates the correct value by at most an additive term of $1/2^{n^b}$).*

Proof. Let $h_p^\Pi = (M/p)^{-1} \bmod p$ for each prime $p \in \Pi$.

If we were to compute a good approximation $\tilde{A}_\Pi$ to the fractional part of:

$$A_\Pi = \sum_{p\in\Pi} \frac{(2^{2(n+1)t} W(X,Y) \bmod p) h_p^\Pi}{p}$$

i.e. if $\tilde{A}_\Pi$ were a good approximation to $A_\Pi - \lfloor A_\Pi \rfloor$, then $\tilde{A_\Pi} M$ would be a good approximation to $2^{2(n+1)t} W(X,Y)$. This follows from observing that the fractional part of A_Π is exactly $\frac{2^{2(n+1)t}W(X,Y)}{M}$ (as in [12,3]).

Instead, we will compute a good approximation $\widetilde{A'}_\Pi$ to the fractional part of

$$A'_\Pi = \sum_{p\in\Pi} \sum_{j=0}^{2n+1} \frac{(W_j(X,Y) \bmod p) h_p^\Pi}{p}.$$

Note: the two quantities A_Π, A'_Π are not equal but their *fractional parts* are. Since we are adding $2(n+1)|\Pi|$ approximate quantities it suffices to compute each of them to $b_m = 2n^b + 2(n+1)|\Pi|$ bits of accuracy to ensure:

$$0 \leq \frac{W(X,Y)}{M} - \widetilde{A'}_\Pi < \frac{1}{2^{n^b}}.$$

Now we analyze the complexity. By Lemma 1, we obtain in $\widehat{\mathsf{TC}}^0_1$ the CRR_Π representation of $W_j(X,Y) \in [0, 2^n)$ for $j \in \{0, \ldots, O(n)\}$. Each h_p^Π is computable in $\widehat{\mathsf{TC}}^0_1$, and poly-many bits of the binary expansion of $1/p$ can be obtained in AC^0. Using AC^0 circuitry we multiply together the $O(\log n)$-bit numbers

h_p^{Π} and $W_j(X,Y) \bmod p$, to obtain the binary expansion of $((W_j(X,Y) \bmod p)h_p^{\Pi}) \cdot (1/p)$ (since multiplying an n-bit number by a $\log n$ bit number can be done in AC^0). Thus, with one more layer of majority gates, we compute

$$A'_{\Pi} = \sum_{p \in \Pi} \sum_{j=0}^{2n+1} \frac{(W_j(X,Y) \bmod p)h_p^{\Pi}}{p}$$

and strip off the integer part, to obtain the desired approximation. □

Corollary 1. *Let Π be any set of primes such that the product M of these primes lies in $(2^{n^c}, 2^{n^d})$ for a fixed constant $d > c \geq 3$. Then, given Z in CRR_{Π} representation and the numbers h_p^{Π} for each $p \in \Pi$, we can compute the binary representation of a* good *approximation to $\frac{Z}{M}$ in $\widehat{\mathsf{TC}}_1^0$*

Now, finally, we present our desired approximation. $\widetilde{V}(X,Y)$ is $2^{n^c} \cdot V'(X,Y)$, where $V'(X,Y)$ is an approximation (within $1/2^{n^{2c}}$) of

$$V(X,Y) = \frac{W(X,Y)\prod_{i=1}^{n^c}(M_i - 1)/2}{\prod_{i=1}^{n^c} M_i}.$$

$$\begin{aligned} W(X,Y) - 2^{n^c}V(X,Y) &= W(X,Y) - 2^{n^c}\frac{W(X,Y)\prod_{i=1}^{n^c}(M_i-1)/2}{\prod_{i=1}^{n^c} M_i} \\ &= W(X,Y) - \frac{W(X,Y)\prod_{i=1}^{n^c}(M_i-1)}{\prod_{i=1}^{n^c} M_i} \\ &< W(X,Y)\frac{n^c}{2^{n^c}} < \frac{2^{2n}n^c}{2^{n^c}} \end{aligned}$$

and

$$\begin{aligned} 2^{n^c}V(X,Y) - \widetilde{V}(X,Y) &= 2^{n^c}V(X,Y) - 2^{n^c}V'(X,Y) \\ &= 2^{n^c}(V(X,Y) - V'(X,Y)) \\ &\leq 2^{n^c}(\frac{1}{2^{n^{2c}}}) = \frac{2^{n^c}}{2^{n^{2c}}}. \end{aligned}$$

Thus $X/Y - \widetilde{V}(X,Y) = (X/Y - W(X,Y) + (W(X,Y) - 2^{n^c}V(X,Y)) + (2^{n^c}V(X,Y) - \widetilde{V}(X,Y)) < 2^{-(n+1)} + n^c 2^{2n}/2^{n^c} + 2^{n^c}/2^{n^{2c}} < 1/Y$.

Lemma 3. *Let Π_i for $i \in \{1, \ldots, n^c\}$ be n^c pairwise disjoint sets of primes such that $M_i = \prod_{p \in \Pi_i} p \in (2^{n^c}, 2^{n^d})$ (for some constants $c, d : 3 \leq c < d$). Let $\Pi = \cup_{i=1}^{n^c} \Pi_i$. Then, given X, Y and the Π_i, we can compute $\widetilde{V}(X,Y)$ in $\widehat{\mathsf{TC}}_3^0$.*

Proof. In $\widehat{\mathsf{TC}}_1^0$ we compute the CRR_{Π} representation of each M_i, and the numbers $W_j \bmod p$ (using Lemma 1). Also, as in Lemma 2, we get the values h_p^{Π}.

Then, with one more layer of majority gates we can compute the CRR representation of $\prod_i (M_i - 1)/2$ and of $2^{2(n+1)t}W(X,Y) = \sum_{j=0}^{2n+1} W_j(X,Y)$. The CRR representation of the product $2^{2(n+1)t}W(X,Y) \cdot \prod_i (M_i - 1)/2$ can then be computed with AC^0 circuitry to obtain the CRR representation of the numerator of the expression for $V(X,Y)$. (It is important to note that $2^{2(n+1)t}W(X,Y) \cdot \prod_i (M_i - 1)/2 < \prod_i M_i$, so that it is appropriate to talk about this CRR representation. Indeed, that is the reason why we divide each factor $M_i - 1$ by two.)

This value can then be converted to binary with one additional layer of majority gates, via Corollary 1, to obtain $\widetilde{V}(X,Y)$. □

This completes the proof of Theorem 1. □

Corollary 2. *Let Π be any set of primes such that the product M of these primes lies in $(2^{n^c}, 2^{n^d})$ for a fixed constant $d > c \geq 3$. Then, given Z in CRR$_\Pi$ representation, the binary representation of Z can be computed in $\widehat{\mathsf{TC}}^0_3$*

Proof. Recall from the proof of Theorem 1 that, in order to compute the bits of $Z/2$, our circuit actually computes an approximation to $(Z+1)/2$. Although, of course, it is trivial to compute $Z/2$ if Z is given to us in binary, let us consider how to modify the circuit described in the proof of Lemma 3, if we were computing $\widetilde{V}(Z+1,2)$, where we are given Z in CRR representation.

With one layer of majority gates, we can compute the CRR$_\Pi$ representation of each M_i and the values h_p^Π for each prime p. (We will not need the numbers $W_j \bmod p$.)

Then, with one more layer of majority gates we can compute the CRR representation of $\prod_i (M_i - 1)/2$. In place of the gates that store the value of the CRR representation of $2^{2(n+1)t}W(X,Y)$, we insert the CRR representation of Z (which is given to us as input) and using AC^0 circuitry store the value of $Z+1$. The CRR representation of the product $Z + 1 \cdot \prod_i (M_i - 1)/2$ can then be computed with AC^0 circuitry to obtain the CRR representation of the numerator of the expression for $V(Z+1,2)$.

Then this value can be converted to binary with one additional layer of majority gates, from which the bits of Z can be read off. □

It is rather frustrating to observe that the input values Z are not used until quite late in the $\widehat{\mathsf{TC}}^0_3$ computation (when just one layer of majority gates remains). However, we see no simpler uniform algorithm to convert CRR to binary.

For our application regarding problems in the counting hierarchy, it is useful to consider the analog to Theorem 1 where the values X and Y are presented in CRR notation.

Theorem 2. *The function taking as input $X \in [0, 2^n), Y \in [1, 2^n)$ (in CRR) as well as 0^m, and producing as output the binary expansion of X/Y correct to m places is in $\widehat{\mathsf{TC}}^0_3$.*

Proof. We assume that the CRR basis consists of pairwise disjoint sets of primes M_i, as in Lemma 3.

The algorithm is much the same as in Theorem 1, but there are some important differences that require comment. The first step is to determine if $Y = 1$, which can be done using AC^0 circuitry (since the CRR of 1 is easy to recognize). The next step is to determine a value t such that $2^{t-1} \leq Y < 2^t$. Although this is trivial when the input is presented in binary, when the input is given in CRR it requires the following lemma:

Lemma 4. *(Adapted from [1,7,3]) Let X be an integer from $(-2^n, 2^n)$ specified by its residues modulo each $p \in \Pi_n$. Then, the predicate $X > 0$ is in $\widehat{\mathsf{TC}}^0_2$*

Since we are able to determine inequalities in majority-depth two, we will carry out the initial part of the algorithm from Theorem 1 using *all* possible values of t, and then select the correct value between the second and third levels of MAJORITY gates.

Thus, for each t, and for each j, we compute the values $W_{j,t}(X+1, Y) = (X+1)(2^t - Y)^j(2^{(2n+1-j)t})$ in CRR, along with the desired number of bits of accuracy of $1/p$ for each p in our CRR basis.

With this information available, as in Lemma 3, in majority-depth one we can compute h_p^{Π}, as well as the CRR representation of each M_i, and thus with AC^0 circuitry we obtain $(W_{j,t}(X+1, Y)$ and the CRR for each $(M_i - 1)/2$.

Next, with our second layer of majority gates we sum the values $W_{j,t}(X + 1, Y)$ (over all j), and at this point we also will have been able to determine which is the correct value of t, so that we can take the correct sum, to obtain $2^{2(n+1)t}W(X, Y)$.

Thus, after majority-depth two, we have obtained the same partial results as in the proof of Lemma 3, and the rest of the algorithm is thus identical. □

Proposition 2. *Iterated product is in uniform $\widehat{\mathsf{TC}}^0_3$.*

Proof. The overall algorithm is identical to the algorithm outlined in [20], although the implementation of the basic building blocks is different. In majority-depth one, we convert the input from binary to CRR. With one more level of majority gates, we compute the CRR of the product.

Simultaneously, in majority-depth two we compute the bottom two levels of our circuit that computes from CRR to binary, as in Corollary 2.

Thus, with one final level of majority gates, we convert the answer from CRR to binary. □

3.1 Consequences for the Counting Hierarchy

Corollary 3. $\mathsf{BitSLP} \in \mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$.

Proof. This is immediate from Proposition 1 and Corollary 2.

Let f be the function that takes as input a tuple $(C, (p, j))$ and if p is a prime, evaluates the arithmetic circuit C mod p and outputs the j-th bit of the

result. This function f, taken together with the $\widehat{\mathsf{TC}}^0_3$ circuit family promised by Corollary 2, satisfies the hypothesis of Proposition 1. (There is a minor subtlety, regarding how to partition the set of primes into the groupings M_i, but this is easily handled by merely using all of the primes of a given length, at most polynomially-larger than $|C|$.) □

Via essentially identical methods, using Theorem 2, we obtain:

Corollary 4. *$\{(C_X, C_Y, i)$: the i-th bit of the quotient X/Y, where X and Y are represented by arithmetic circuits C_X and C_Y, respectively, is in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$.*

4 Integer Matrix Powering

Theorem 3. *The function $MPOW(A, m, p, q, i)$ taking as input a $(d \times d)$ integer matrix $A \in \{0,1\}^{d^2 n}$, p, q, 1^i, where $p, q \in [d]$, $i \in [O(n)]$ and producing as output the i-th bit of the (p, q)-th entry of A^m is in $\widehat{\mathsf{TC}}^0_3$.*

For a $(d \times d)$ matrix, the characteristic polynomial $\chi_A(x) : \mathbb{Z} \to \mathbb{Z}$ is a univariate polynomial of degree at most d. Let $q, r : \mathbb{Z} \to \mathbb{Z}$ be univariate polynomials of degree at most $(m - d)$ and $(d - 1)$ such that $x^m = q(x)\chi_A(x) + r(x)$. By the Cayley-Hamilton theorem, we have that $\chi_A(A) = 0$. So, in order to compute A^m, we just have to compute $r(A)$.

Lemma 5. *Given a $(d \times d)$ matrix A of n-bit integers, the coefficients of the characteristic polynomial of A in CRR can be computed in $\widehat{\mathsf{TC}}^0_1$.*

Proof. We convert the entries of A to CRR and compute the determinant of $(xI - A)$. This involves an iterated sum of $O(2^d d!)$ integers each of which is an iterated product of d n-bit integers. The conversion to CRR is in $\widehat{\mathsf{TC}}^0_1$. Since addition, multiplication, and powering of $O(1)$ numbers of $O(\log n)$ bits is computable in AC^0, it follows that the coefficients of the characteristic polynomial can be computed in $\widehat{\mathsf{TC}}^0_1$.

Lemma 6. *Given the coefficients of the polynomial r, in CRR, and given A in CRR, we can compute A^m in CRR using AC^0 circuitry.*

Proof. Recall that $A^m = r(A)$. Let $r(x) = r_0 + r_1 x + \ldots + r_{d-1} x^{d-1}$. Computing any entry of $r(A)$ in CRR involves an iterated sum of $O(1)$ many numbers which are themselves an iterated product of $O(1)$ many $O(\log n)$-bit integers. □

Lemma 7. *(Adapted from [11]) Let p be a prime of magnitude $\mathsf{poly}(m)$. Let $g(x)$ of degree m and $f(x)$ of degree d be monic univariate polynomials over GF_p, such that $g(x) = q(x)f(x) + r(x)$ for some polynomials $q(x)$ of degree $(m - d)$ and $r(x)$ of degree $(d-1)$. Then, given the coefficients of g and f, the coefficients of r can be computed in $\widehat{\mathsf{TC}}^0_1$.*

Proof. Following [11], let $f(x) = \sum_{i=0}^{d} a_i x^i$, $g(x) = \sum_{i=0}^{m} b_i x^i$, $r(x) = \sum_{i=0}^{d-1} r_i x^i$ and $q(x) = \sum_{i=0}^{m-d} q_i x^i$. Since f, g are monic, we have $a_d = b_m = 1$. Denote by $f_R(x), g_R(x), r_R(x)$ and $q_R(x)$ respectively the polynomial with the i-th coefficient $a_{d-i}, b_{m-i}, r_{d-i-1}$ and q_{m-d-i} respectively. Then note that $x^d f(1/x) = f_R(x)$, $x^m g(1/x) = g_R(x)$, $x^{m-d} q(1/x) = q_R(x)$ and $x^{d-1} r(1/x) = r_R(x)$.

We use the Kung-Sieveking algorithm (as implemented in [11]). The algorithm is as follows:

1. Compute $\tilde{f}_R(x) = \sum_{i=0}^{m-d} (1 - f_R(x))^i$ via interpolation modulo p.
2. Compute $h(x) = \tilde{f}_R(x) g_R(x) = c_0 + c_1 x + \ldots + c_{d(m-d)+m} x^{d(m-d)+m}$. from which the coefficients of $q(x)$ can be obtained as $q_i = c_{d(m-d)+m-i}$.
3. Compute $r(x) = g(x) - q(x) f(x)$.

To prove the correctness of our algorithm, note that we have $g(1/x) = q(1/x) f(1/x) + r(1/x)$. Scaling the whole equation by x^m, we get $g_R(x) = q_R(x) f_R(x) + x^{m-d+1} r_R(x)$. Hence when we compute $h(x) = \tilde{f}_R(x) g_R(x)$ in step 2 of our algorithm, we get

$$h(x) = \tilde{f}_R(x) g_R(x) = \tilde{f}_R(x) q_R(x) f_R(x) + x^{m-d+1} \tilde{f}_R(x) r_R(x).$$

Note that $\tilde{f}_R(x) f_R(x) = \tilde{f}_R(x)(1 - (1 - f_R(x))) = \sum_{i=0}^{m-d} (1 - f_R(x))^i - \sum_{i=0}^{m-d} (1 - f_R(x))^{i+1} = 1 - (1 - f_R(x))^{m-d+1}$ (a telescoping sum). Since f is monic, f_R has a constant term which is 1 and hence $(1 - f_R(x))^{m-d+1}$ does not contain a monomial of degree less than $(m - d + 1)$. This is also the case with $x^{m-d+1} \tilde{f}_R(x) r_R(x)$, and hence all the monomials of degree less than $(m - d + 1)$ belong to $q_R(x)$.

To see that this can be done in $\widehat{\mathsf{TC}}^0_1$, first note that given $f(x)$ and $g(x)$, the coefficients of $f_R(x)$ and $g_R(x)$ can be computed in NC^0. To compute the coefficients of $\tilde{f}_R(x)$, we use interpolation via the discrete Fourier transform (DFT) using arithmetic modulo p. Find a generator w of the multiplicative group modulo p and substitute $x = \{w^1, w^2, \ldots, w^{p-1}\}$ to obtain a system of linear equations in the coefficients F of $\tilde{f}_R(x) : V \cdot F = Y$, where Y is the vector consisting of $\tilde{f}_R(w^i)$ evaluated at the various powers of w. Since the underlying linear transformation $V(w)$ is a DFT, it is invertible; the inverse DFT $V^{-1}(w)$ is equal to $V(w^{-1}) \cdot (p-1)^{-1}$, which is equivalent to $-V(w^{-1}) \bmod p$. We can find each coefficient of $\tilde{f}_R(x)$ evaluating $V^{-1}Y$, i.e., by an inner product of a row of the inverse DFT-matrix with the vector formed by evaluating $\sum_{i=1}^{(m-d+1)} (1 - f_R(x))^{i-1}$ at various powers of w and dividing by $p - 1$. The terms in this sum can be computed in AC^0, and then the sum can be computed in majority-depth one, to obtain the coefficients of $\tilde{f}_R(x)$. The coefficients of $h(x)$ in step 2 could be obtained by iterated addition of the product of certain coefficients of $\tilde{f}_R$ and g_R, but since the coefficients of $\tilde{f}_R$ are themselves obtained by iterated addition of certain terms t, we roll steps 1 and 2 together by multiplying these terms t by the appropriate coefficients of g_R. Thus steps 1 and 2 can be accomplished in majority-depth 1. Then step 3 can be computed using AC^0 circuitry. □

Proof. (of Theorem 3)

Our $\widehat{\mathsf{TC}}^0_3$ circuit C that implements the ideas above is the following:

0. At the input, we have the d^2 entries A_{ij}, $i, j \in [d]$ of A, and a set Π of short primes (such that Π can be partitioned in to n^c sets Π_i that are pairwise disjoint, i.e., $\Pi = \cup_{i=1}^{n^c} \Pi_i$).
1. In majority-depth one, we obtain (1) $A_{ij} \mod p$ for each prime p in our basis, and (2) $M_i = \prod_{p \in \Pi_i} p$ for all the n^c sets that constitute Π, and (3) the CRR of the characteristic polynomial of A (via appeal to Lemma 5).
2. In the next layer of threshold gates, we compute (1) $\prod_i^{n^c} (M_i - 1)/2$ in CRR, and (2) the coefficients of the polynomial r in CRR, by appeal to Lemma 7.
3. At this point, by Lemma 6, AC^0 circuitry can obtain $r(A) = A^m$ in CRR, and with one more layer of MAJORITY gates we can convert to binary, by appeal to Corollary 2.

□

5 Open Questions and Discussion

Is conversion from CRR to binary in dlogtime-uniform $\widehat{\mathsf{TC}}^0_1$*?* This problem has been known to be in P-uniform $\widehat{\mathsf{TC}}^0_1$ starting with the seminal work of Beame, Cook, and Hoover [4], but the subsequent improvements on the uniformity condition [5,12] introduced additional complexity that translated into increased depth. We have been able to reduce the majority-depth by rearranging the algorithmic components introduced in this line of research, but it appears to us that a fresh approach will be needed, in order to decrease the depth further.

Is BitSLP *in* $\mathsf{PH}^{\mathsf{PP}}$*?* An affirmative answer to the first question implies an affirmative answer to the second, and this would pin down the complexity of BitSLP between $\mathsf{P}^{\#\mathsf{P}}$ and $\mathsf{PH}^{\mathsf{PP}}$.

Is PosSLP *in* PH*?* Some interesting observations related to this problem were announced recently [8,16].

Is it easy to compute bits of large powers of small matrices? Recall that some surprising things about large powers of *integers* can be computed [13].

Acknowledgments

The first author acknowledges the support of NSF grants CCF-0832787 and CCF-1064785.

References

1. Agrawal, M., Allender, E., Datta, S.: On TC^0, AC^0, and Arithmetic circuits. Journal of Computer and System Sciences 60(2), 395–421 (2000)
2. Allender, E., Schnorr, H.: The complexity of the BitSLP problem (2005), unpublished manuscript

3. Allender, E., Bürgisser, P., Kjeldgaard-Pedersen, J., Miltersen, P.B.: On the complexity of numerical analysis. SIAM J. Comput. 38(5), 1987–2006 (2009)
4. Beame, P.W., Cook, S.A., Hoover, H.J.: Log depth circuits for division and related problems. SIAM Journal on Computing 15, 994–1003 (1986)
5. Chiu, A., Davida, G.I., Litow, B.: Division in logspace-uniform NC^1. Informatique Théorique et Applications 35(3), 259–275 (2001)
6. Datta, S., Pratap, R.: Computing bits of algebraic numbers. In: TAMC. pp. 189–201 (2012)
7. Dietz, P., Macarie, I., Seiferas, J.: Bits and relative order from residues, space efficiently. Information Processing Letters 50(3), 123–127 (1994)
8. Etessami, K.: Probability, recursion, games, and fixed points (2013), talk presented at Horizons in TCS: A Celebration of Mihalis Yannakakis' 60th Birthday
9. Etessami, K., Yannakakis, M.: On the complexity of Nash equilibria and other fixed points. SIAM J. Comput. 39(6), 2531–2597 (2010)
10. Goldmann, M., Karpinski, M.: Simulating threshold circuits by majority circuits. SIAM J. Comput. 27(1), 230–246 (1998)
11. Healy, A., Viola, E.: Constant-depth circuits for arithmetic in finite fields of characteristic two. In: STACS 2006, pp. 672–683. Springer (2006)
12. Hesse, W., Allender, E., Barrington, D.: Uniform constant-depth threshold circuits for division and iterated multiplication. Journal of Computer and System Sciences 65, 695–716 (2002)
13. Hirvensalo, M., Karhumäki, J., Rabinovich, A.: Computing partial information out of intractable: Powers of algebraic numbers as an example. Journal of Number Theory 130, 232–253 (2010)
14. Hunter, P., Bouyer, P., Markey, N., Ouaknine, J., Worrell, J.: Computing rational radical sums in uniform TC^0. In: FSTTCS. pp. 308–316 (2010)
15. Jeřábek, E.: Root finding with threshold circuits. Theoretical Computer Science 462, 59–69 (2012)
16. Jindal, G., Saranurak, T.: Subtraction makes computing integers faster. CoRR abs/1212.2549 (2012)
17. Kayal, N., Saha, C.: On the sum of square roots of polynomials and related problems. TOCT 4(4), 9 (2012)
18. Koiran, P., Perifel, S.: The complexity of two problems on arithmetic circuits. Theor. Comput. Sci. 389(1-2), 172–181 (2007)
19. Koiran, P., Perifel, S.: Interpolation in Valiant's theory. Computational Complexity 20(1), 1–20 (2011)
20. Maciel, A., Thérien, D.: Threshold circuits of small majority-depth. Inf. Comput. 146(1), 55–83 (1998)
21. Mereghetti, C., Palano, B.: Threshold circuits for iterated matrix product and powering. ITA 34(1), 39–46 (2000)
22. Ouaknine, J., Worrell, J.: Positivity problems for low-order linear recurrence sequences. In: SODA. pp. 366–379 (2014)
23. Sherstov, A.A.: Powering requires threshold depth 3. Inf. Process. Lett. 102(2-3), 104–107 (2007)
24. Siu, K.Y., Roychowdhury, V.P.: On optimal depth threshold circuits for multiplication and related problems. SIAM J. Discrete Math. 7(2), 284–292 (1994)
25. Toda, S.: PP is as hard as the polynomial time hierarchy. SIAM J. Comput. 20, 865–877 (1991)
26. Vollmer, H.: Introduction to Circuit Complexity. Springer-Verlag (1999)
27. Wegener, I.: Optimal lower bounds on the depth of polynomial-size threshold circuits for some arithmetic functions. Inf. Process. Lett. 46(2), 85–87 (1993)