Uniform Valiant's Classes

August 10, 2018

Contents

Some Preliminary Definitions and Lemmata

Arithmetic Complexity

We'll need a well-defined representation of an arithmetic circuit soon, so let's define it here, following burgisser_algebraic-complexity-theory [Bür00]

def:arithcirc **Definition 1.**

- An **arithmetic circuit** Γ over a field $\mathbb{F}$ with m inputs and **size** r is a sequence $(\Gamma_1, \ldots, \Gamma_r)$ of instructions $\Gamma_\rho = (\omega_\rho \colon i_\rho, j_\rho)$ with operation symbols $\omega_\rho \in \{+, \times\}$ and addresses $i_\rho, j_\rho \in \mathbb{Z}$ satisfying $-m < i_\rho, j_\rho < \rho$ (the first m addresses are the inputs, the next r are the gates)

- An arithmetic circuit Γ defines a directed acyclic multigraph, with nodes $\{\rho \in \mathbb{Z} \colon -m < \rho \leq r\}$ and edges $(i_\rho, \rho), (j_\rho, \rho)$.
- For any sequence of input polynomials $a_1, \ldots, a_m$, Γ has a **result sequence** $(b_{-m+1}, \ldots, b_r)$ defined by $b_\rho = a_{m+\rho}$ for $\rho \leq 0$ (corresponding to the input variables) and $b_\rho = b_{i_\rho} \omega_\rho b_{j_\rho}$ for $\rho > 0$ (corresponding to operation gates).

fournier_fixed-polynomial_2013

The following two definitions are lifted almost verbatim from [FPV13]

Definition 2. Let $\mathbb{F}$ be a field. If $s \colon \mathbb{N} \to \mathbb{N}$ is a function, a family $\{P_n\}$ of polynomials over $\mathbb{F}$ is in $\mathsf{asize}_{\mathbb{F}}(s(n))$ if it is computed by a family of arithmetic circuits of size $O(s(n))$ over $\mathbb{F}$

Note that the only difference between $\bigcup_k \mathsf{asize}_{\mathbb{F}}(n^k)$ and VP is that we require the formal degree of families in VP to be polynomially bounded.

Definition 3. Let C be a complexity class. A family of circuits $\{\mathcal{C}_n\}$ is called **C-uniform** if the encoding of $\mathcal{C}_n$ can be computed by an algorithm in C. A family of polynomials $\{P_n\}$ over a field $\mathbb{F}$ is in the class $\mathsf{unif}_{\mathsf{C}}\text{-}\mathsf{VP}_{\mathbb{F}}$ if it is computed by a C-uniform family of constant-free arithmetic circuits of polynomial formal degree.

A family of polynomials $\{Q_n(x)\}$ over a field $\mathbb{F}$ is in the class $\mathsf{unif}_{\mathsf{C}}\text{-}\mathsf{VNP}_{\mathbb{F}}$ if Q_n has n variables $x_1, \ldots, x_n$ and there exists a family $\{P_n(x,y)\} \in \mathsf{unif})\mathsf{C}\text{-}\mathsf{VP}_{\mathbb{F}}$ such that

$$Q_n(x) = \sum_{y \in \{0,1\}^{m(n)}} P_n(x,y)$$

where $m(n)$ denotes the length of y in P_n

burgisser_algebraic-complexity-theory

Definition 4 ([Bür00])**.** Let $\{P_n\}$ be a family of polynomials with n variables and polynomial size circuits. A function $\{\phi_n\}$ is called a **Boolean part** of $\{P_n\}$ if we have, over some field of characteristic 0, $m(n) = n^{O(1)}$ and

$$\forall x \in \{0,1\}^n \colon f_n 9x) = \sum_{i=1}^{m(n)} \phi_{n,i}(x) 2^{i-1}$$

Then, the Boolean part of a class of polynomials is the set of Boolean parts of polynomial families in that class.

Definition 5. The **weight** of a polynomial is the sum of the absolute values of its coefficients

fournier_fixed-polynomial_2013

lm:weight **Lemma 6** ([FPV13])**.** *Consider a polynomial with size s and degree d with constants of absolute value absolutely bounded by $M \geq 2$. Its weight is bounded by M^{sd}*

koiran_hilberts_1996

lm:grhlem **Lemma 7** ([Koi96])**.** *Let S be a system of polynomial equations $P_1(x) = 0, \ldots, P_m(x) = 0$ with coefficients in $\mathbb{Z}$, n unknowns, and for all i, degree of P_i at most d and weight of P_i at most w. If the system S has a solution over $\mathbb{C}$ then under the Generalized Riemann Hypothesis,*

$$\pi_S(a) \geq \frac{\pi(a)}{d^{O(n)}} - \sqrt{a}\log(wa)$$

where $\pi(a)$ denotes the number of primes at most a and $\pi_S(a)$ denotes the number of primes p such that S has a solution over $\mathbb{F}_p$

Boolean Complexity

def:arithcirc

def:boolcirc **Definition 8.** **Boolean circuits** are presented just as in Definition 1, except with $\omega_\rho = \{\wedge, \vee\}$ and allowing $\neg$ gates on the addresses

Definition 9. FNC^i consists of string functions computable by logspace-uniform Boolean circuit families of depth $O(\log^i n)$ and polynomial size. $\mathsf{FNC}^i/\mathsf{poly}$ consists of string functions computable by (not necessarily uniform) Boolean circuit families of depth $O(\log^i n)$ and polynomial size.

vinodchandran_pp-circuit-complexity

lm:pp-fixedpoly **Lemma 10** ([V V05])**.** *For all $k \in \mathbb{N}$,*

$$\mathsf{PP} \not\subseteq \mathsf{size}[n^k]$$

An Unconditional Fixed-Polynomial Size Circuit Lower Bound for P-Uniform VNP

fournier_fixed-polynomial_2013

lm:valcrit-p **Lemma 11** (P-Uniform Valiant's Criterion, per [FPV13])**.** *In characteristic zero, a family $\{P_n\}$ is in $\mathsf{unif_P}$-VNP if and only if P_n has n variables, a polynomial degree, and its coefficients are computable in GapP; that is, the function mapping $(c_1, \ldots, c_n)$ to the coefficient of $X_1^{c_1} \ldots X_n^{c_n}$ in P_n is in GapP.*

rm:c-notation *Remark* 1. Note that here, c is given to ϕ in binary representation, as functions in GapP are maps from $\{0,1\}^* \to \mathbb{N}$. For the sake of this discussion, let's take the length of c to be $n' = n \lceil \log \deg(P_n) \rceil$. Thus, c is a series of n blocks of bits, each encoding an integer at most the degree of P_n. Furthermore, let c_i denote the integer encoded by the i^{th} block of bits of c, and c_{ij} to be the j^{th} bit of the i^{th} block of bits.

thm:p-unifvnp-lb **Theorem 12.** *Assume GRH. For all $k \in \mathbb{N}$, $\mathsf{unif}_{\mathsf{P}}\text{-}\mathsf{VNP} \not\subseteq \mathsf{asize}_{\mathbb{C}}(n^k)$*

fournier_fixed-polynomial_2013

burgisser_algebraic-complexity-theory

Proof. We will follow the approach presented in [FPV13], which follows that of [Bür00]. Suppose $\mathsf{unif}_{\mathsf{P}}\text{-}\mathsf{VNP}_{\mathbb{F}} \subseteq \mathsf{asize}_{\mathbb{C}}(n^k)$, and let L be a language in PP. Then, there exists a polynomial-time verifier $\phi(x,y)\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, for $m = n^{O(1)}$, such that

$$x \in L \iff \text{there are at least } 2^{m-1}+1 \; y \in \{0,1\}^m \text{ such that } \phi(x,y) = 1$$

Define the polynomial

$$P_n(X) = \sum_{x\in\{0,1\}^n} \sum_{y \in \{0,1\}^m} \phi(x,y) \prod_i^n X_i^{x_i}(1-X_i)^{1-x_i}$$

Then, if we evaluate P_n at $X = x$, the only term on the right-hand side that survives has value

$$\sum_{y\in\{0,1\}^m} \phi(x,y)$$

and thus

$$x \in L \iff P_n(x) > 2^{m-1}$$

The sum

$$f(x) = \sum_{y\in\{0,1\}^m} \phi(x,y)$$

lm:valcrit-p

is in GapP, so by Lemma 11, $\{P_n(X)\} \in \mathsf{unif}_{\mathsf{P}}\text{-}\mathsf{VNP}_{\mathbb{F}}$. By assumption, this means that for some k, $\{P_n(X)\} \in \mathsf{asize}_{\mathbb{C}}(n^k)$. Our goal now will be to show that there exists a Boolean circuit - also testing for membership in L - of size $O(n^{k'})$ for a constant k' depending only on k (in particular, not on L). This would yield a contradiction, as we know $\mathsf{PP} \not\subset \mathsf{SIZE}[n^k]$. Our approach will be to replace the complex constants in the original circuits with constants that allow us to compute the same polynomial over $\mathbb{F}_p$.

Let $\alpha = \alpha_1, \ldots, \alpha_t$ be the set of complex constants used in the circuit. Define the polynomial $C_n(X, Z)$ such that

$$C_n(X, \alpha) = P_n(X)$$

where we've replaced each of the constants in the original polynomial with a variable. Because the size of the initial circuit was $O(n^k)$ by assumption, the number of variables in C_n is $O(n^k)$ as well. Consider the following system of equations

$$\forall x \in L, C_n(x, Z) = r_x$$

where $r_x = \sum_{y \in \{0,1\}^m} \phi(x, y) \in \{0, 1, \ldots, 2^m\}$. The degree of each polynomial in the system is the same as the degree of C_n, which is at most $2^t = 2^{O(n^k)}$. Every coefficient appearing is bounded by $2^m = 2^{O(n^k)}$ (because each is either a counting problem or a value r_x) and we know from Lemma 6 that this means that the weight is at most $(2^m)^{2^{O(n^k)}} = 2^{2^{O(n^k)}}$.

Now, we're in a position where we can apply Lemma 7. We know the system is solvable over $\mathbb{C}$ because a solution corresponds to a polynomial that can decide L and we know P_n works. Applying the lemma with $d = 2^{O(n^k)}, w = 2^{2^{O(n^k)}}$, and using the Prime Number Theorem, we find that the system has a solution over $\mathbb{F}_p$ for $p = 2^{O(n^{2k})}$. But if the system has a solution over $\mathbb{F}_p$, that means that there's a choice of $Z \in \mathbb{F}_p^t$ such that $C_n(X, Z)$ is a decider for L when C_n is evaluated over $\mathbb{F}_p$.

Thus, we've replaced our polynomial-size arithmetic circuit over $\mathbb{C}$ with a polynomial-size arithmetic circuit over $\mathbb{F}_p^t$ for $p = 2^{O(n^{2k})}$. But now that we're in a finite field, we can replace all $+$ and $\times$ gates with Boolean operations, using $\log_2 p$ bits to represent an element of $\mathbb{F}_p$, and $O(\log^2 p)$ gates to simulate an arithmetic operation. Thus, we have Boolean circuits of size $n^{O(k)}$ to decide $L \in \mathsf{PP}$. But this contradicts Lemma 10, and so we're done!

□

$\mathsf{unif}_\mathsf{L}\text{-}\mathsf{VNP}_\mathbb{F} = \mathsf{unif}_\mathsf{P}\text{-}\mathsf{VNP}_\mathbb{F}$

lm:l-uniform-verif **Lemma 13.** *Consider a polynomial-time verifier $\mathcal{V}(X, Y)$. There exists a polynomial family $\{f_n(X_1, \ldots, X_n, Z_1, \ldots, Z_{u(n)})\} \in \mathsf{unif}_\mathsf{L}\text{-}\mathsf{VP}_\mathbb{F}$ computable in L and a polynomial $u(n)$ such that*

$$\sum_{y \in \{0,1\}^m} \mathcal{V}(X_1, \ldots, X_n, y) = \sum_{z \in \{0,1\}^{u(n)}} f_n(X, z)$$

Proof. We'll first show that such a polynomial family exists. We'll then argue that it can be computed in L.

We'll show existence constructively. Start by converting $\mathcal{V}(X_1, \ldots, X_n, Y)$ into a size $(n+m)^k$ circuit on $(n+m)$ inputs for some constant k (this can be done in L [AB09]). Then, convert each gate of $\mathcal{C}$ into a Boolean 3CNF-formula χ_n on $n + s(n)$ variables as follows

- The gate $a = b \wedge c$ is replaced with the formula

$$(\overline{x_a} \vee x_b \vee x_c) \wedge (\overline{x_a} \vee \overline{x_b} \vee x_c) \wedge (\overline{x_a} \vee x_b \vee \overline{x_c}) \wedge (x_a \vee \overline{x_b} \vee \overline{x_c})$$

- The gate $a = b \vee c$ is replaced with the formula

$$(\overline{x_a} \vee x_b \vee x_c) \wedge (x_a \vee x_b \vee \overline{x_c}) \wedge (x_a \vee \overline{x_b} \vee x_c) \wedge (x_a \vee \overline{x_b} \vee \overline{x_c})$$

- The gate $a = \neg b$ is replaced with the formula

$$(x_a \vee x_b) \wedge (\overline{x_a} \vee \overline{x_b})$$

Note that every satisfying assignment to χ_n corresponds to an satisfying assignment of value to every gate of the circuit, and thus satisfying assignments to χ_n are in bijection with satisfying inputs to $\mathcal{C}$. Therefore,

$$\sum_{y \in \{0,1\}^m} \mathcal{V}(X_1, \ldots, X_n, y) = \sum_{z \in \{0,1\}^{u(n)}} \chi_n(X, z)$$

Now, we can arithmetize the formula $\chi_n(X, z)$ to obtain a polynomial $f_n(X, Z)$ such that

$$\chi_n(x, z) = 1 \iff f_n(x, z) = 1$$

as follows. Proceeding clause by clause,

- $\overline{x_a}$ is replaced with $1 - x_a$
- $x_a \vee x_b$ is replaced with the polynomial $1 - (1 - x_a)(1 - x_b)$

Thus, each clause of χ_n corresponds to a polynomial of degree at most 3, and therefore the entire formula can be represented by a polynomial f_n of degree at most 3 times the number of clauses. Furthermore, it clearly has polynomial-size circuits, since there's $O(1)$ arithmetic gates for each clause of the formula, and the number of clauses is polynomial in n. Therefore, $\{f_n\} \in \mathsf{VP}$.

Now, we must show that we can compute circuits for $\{f_n\}$ in L. The standard L algorithm for outputting a Boolean circuit corresponding to a P algorithm outputs the circuit gate-by-gate, per Definition def:boolcirc 8. This is to say - for gate ρ, it is able to output $(\omega_\rho \colon i_\rho, j_\rho)$, where i_ρ and j_ρ are addresses to the inputs to ρ.

Our algorithm to output the arithmetic circuit for $f_n(X, Y)$ is as follows. Run the L algorithm for computing a circuit. At each stage, store the address of the product of all clauses thusfar. Upon internally computing a Boolean gate, convert the gate to the corresponding product of clause polynomials and output the gates for an arithmetic circuit computing this product (there's only $O(1)$ of everything, so this can be done in logspace). Then, output a multiplication gate connecting this product and the address of the solution thusfar. Update the address of the solution thusfar to that of the newest multiplication gate, and proceed. Because we write down only polynomially many gates, the address of the solution thusfar is $O(\log n)$ and we can store it. $\square$

lm:weak-valcrit-l **Lemma 14** (Weak Valiant's Criterion for unif_L-VNP)**.** *In characteristic zero, if a family $\{P_n\}$ is such that P_n has n variables, a polynomial degree, and is of the form*

$$P_n(X) = \sum_{e \in \{0,1\}^n} \phi(e) X_1^{e_1} X_2^{e_2} \dots X_n^{e_n}$$

where $\phi \in \mathsf{GapP}$, then $\{P_n\} \in \mathsf{unif}_\mathsf{L}$-$\mathsf{VNP}_\mathbb{F}$

Proof. We follow the proof of Valiant's Criterion presented in burgisser_algebraic-complexity-theory [Bür00]. If $\phi \in \mathsf{GapP}$, then there exist a pair of polynomial-time verifiers $\mathcal{V}(x, y), \mathcal{V}'(x, y')$ such that

$$\phi(e) = \sum_{y \in \{0,1\}^{m(n)}} \mathcal{V}(e, y) - \sum_{y \in \{0,1\}^{m'(n)}} \mathcal{V}'(e, y')$$

where $m(n)$ and $m'(n)$ must grow polynomially in n.

Per Lemma lm:l-uniform-verif 13 there exist a pair of L-uniform families of polynomials $\{f_n(X, Y)\}, \{f_n'(X, Y')\}$ such that

$$\phi(e) = \sum_{y \in \{0,1\}^{u(n)}} f_n(e, y) - \sum_{y' \in \{0,1\}^{u'(n)}} f_n'(e, y'),$$

so we can write

$$
\begin{aligned}
P_n(X) &= \sum_{e\in\{0,1\}^n} \left(\sum_{y\in\{0,1\}^{u(n)}} f_n(e,y) - \sum_{y'\in\{0,1\}^{u'(n)}} f'_n(e,y) \right) \prod_{i=1}^{n} X_i^{e_i} \\
&= \sum_{e\in\{0,1\}^n} \left(\sum_{y\in\{0,1\}^{u(n)}} f_n(e,y) - \sum_{y'\in\{0,1\}^{u'(n)}} f'_n(e,y') \right) \prod_{i=1}^{n} (1 + X_i e_i - e_i) \\
&= \sum_{e\in\{0,1\}^n} \sum_{y\in\{0,1\}^u} \sum_{y'\in\{0,1\}^{u'}} \left(f_n(e,y) \prod_{j'=1}^{u'} y'_j - f'_n(e,y') \prod_{j=1}^{u} y_j \right) \prod_{i=1}^{n} (1 + X_i e_i - e_i) \\
&= \sum_{\substack{(e,y,y') \\ \in\{0,1\}^{n+m+m'}}} h_n(X,e,y,y')
\end{aligned}
$$

Observe that $\{h_n(X,E,Y,Y')\}$ is L-uniform because $\{f_n(X,Y)\}$ and $\{f'_n(X,Y')\}$ are L-uniform and we can simply tack on the polynomially-many $O(1)$-size extra multiplicative factors with $\log O\left(n + m(n) + m'(n)\right) = O(\log n)$ space for counting, so we're done! $\square$

lm:valcrit-l **Lemma 15** (Valiant's Criterion for $\mathsf{unif_L}$-VNP)**.** *In characteristic zero, a family $\{P_n\}$ is in $\mathsf{unif_L}$-VNP if and only if P_n has n variables, a polynomial degree, and its coefficients are computable in* GapP*; that is, the function mapping the binary representation of $(c_1,\ldots,c_n) \in \mathbb{N}^n$ to the coefficient of $X_1^{c_1}\ldots X_n^{c_n}$ in P_n is in* GapP.

As earlier, we will use the representation and notation of Remark 1 rm:c-notation

Proof. One direction is easy - if $\{P_n\} \in \mathsf{unif_L}\text{-}\mathsf{VNP}_{\mathbb{F}}$ then $\{P_n\} \in \mathsf{unif_P}\text{-}\mathsf{VNP}_{\mathbb{F}}$, and this direction follows by Lemma 11 lm:valcrit-p

In the other direction, assume that P_n has n variables, a polynomial degree, and is of the form

$$P_n = \sum_{c\in\mathbb{N}^n} \phi(c) \prod_{i=1}^{n} X_i^{c_i}$$

where $\phi(c) \in \mathsf{GapP}$. Define the polynomial family $\{P'_{n'}\}$ as follows. $P'_{n'}$ is a polynomial in the $n'(n) = \lceil n \log \deg P_n \rceil$ variables $\{X_{ij}\}_{\substack{i\le n \\ j\le\lceil\log\deg P_n\rceil}}$. The terms of P_n map to the terms of $P'_{n'}$ as

$$\phi(c)\prod_{i=1}^{n} X_i^{c_i} \leftrightarrow \phi(c)\prod_{i=1}^{n}\prod_{j=1}^{\lceil \log \deg P_n \rceil} X_{ij}^{c_{ij}}$$

Thus, $P'_{n'}$ is multilinear, with terms indexed by $e \in \{0,1\}^{n'}$ and coefficients defined by the same map (the input to ϕ is still a string of bits). $\phi'(e)$ is still in GapP, so per Lemma 14 lm:weak-valcrit-1 we have that $P'_{n'} \in \mathsf{unif_L\text{-}VNP_{\mathbb{F}}}$ and thus can be written as

$$P'_{n'}(X) = \sum_e h'_{n'}(X, e)$$

where $\{h'_{n'}(X, E)\} \in \mathsf{unif_L\text{-}VP_{\mathbb{F}}}$ and $X = \{X_{ij}\}$. Now, to get back to P_n, all we need to do is set $X_{ij} = X_i^{2^j}$ for all $P'_{n'}$. Thus, if $h_n(X, E)$ are the polynomials in the summand resulting from this collapse,

$$P_n(X) = \sum_e h_n(X, e)$$

where $X = \{X_i\}$. All that remains to be shown is that $\{h_n(X, E)\} \in \mathsf{unif_L\text{-}VP_{\mathbb{F}}}$. The degree and size conditions are immediate from the fact that $h'_{n'}(X, E) \in \mathsf{unif_L\text{-}VP_{\mathbb{F}}}$. All we need to show is L-uniformity. Consider the logspace algorithm that outputs an arithmetic circuit $h'_{n'}(X, E)$ gate-by-gate. In this algorithm, replace every instance of X_{ij} with a circuit for $X_i^{2^j}$ right before outputting. Because $j \leq \lceil \log \deg P_n \rceil$, $2^j = O(\deg P_n) = O(n)$. Thus, we can build the circuit for $X_i^{2^j}$ in L and with at most polynomial increase to the size and degree. Thus, in the end, we have a logspace algorithm that outputs the $h_n(X, E)$, as desired. □

Corollary 16.

$$\mathsf{unif_L\text{-}VNP_{\mathbb{F}}} = \mathsf{unif_P\text{-}VNP_{\mathbb{F}}}$$

Corollary 17. *Assume GRH. For all $k \in \mathbb{N}$,* $\mathsf{unif_L\text{-}VNP_{\mathbb{F}}} \not\subseteq \mathsf{asize_{\mathbb{C}}}(n^k)$

Relating Uniform Valiant's Classes and Counting Classes

burgisser_algebraic-complexity-theory [Bür00] proves the following theorem relating the Boolean parts of Valiant's classes to standard complexity classes

Theorem 18. *For $\mathbb{F}$ a characteristic 0 field and under GRH,*

$$\mathsf{FNC}^1/\mathsf{poly} \subseteq \mathsf{BP}(\mathsf{VP}_{\mathbb{F}}) \subseteq \mathsf{FNC}^3/\mathsf{poly}$$

$$\#\mathsf{P}/\mathsf{poly} \subseteq \mathsf{BP}(\mathsf{VNP}_{\mathbb{F}}) \subseteq \mathsf{FP}^{\#\mathsf{P}}/\mathsf{poly}$$

Theorem 19. *For finite fields $\mathbb{F}$ of characteristic p we have*

$$\mathsf{FNC}^1/\mathsf{poly} \subseteq \mathsf{BP}(\mathsf{VP}_{\mathbb{F}}) \subseteq \mathsf{FNC}^2/\mathsf{poly}$$

$$\#_p\mathsf{P}/\mathsf{poly} = \mathsf{BP}(\mathsf{VNP}_{\mathbb{F}})$$

We conjecture the following analog for L-uniform Valiant's classes. The red inclusion is discussed in the last section and the proof if still in progress. All the other parts have tentative proofs.

Theorem 20. *Over a characteristic 0 field,*

$$\mathsf{FNC}^1 \subseteq \mathsf{BP}(\mathsf{unif}_\mathsf{L}\text{-}\mathsf{VP}) \subseteq \mathsf{FNC}^3$$

$$\mathsf{GapP} = \mathsf{BP}(\mathsf{unif}_\mathsf{L}\text{-}\mathsf{VNP})$$

Lemma 21.

$$\mathsf{FNC}^1 \subseteq \mathsf{BP}(\mathsf{unif}_\mathsf{L}\text{-}\mathsf{VP})$$

Proof. We can turn a Boolean circuit with n inputs into an arithmetic circuit that has the same outputs on inputs from $\{0,1\}^n$ in logspace gate-by-gate, as in the proof of Lemma 15. We replace each gate by a constant number of gates, so the resulting arithmetic circuit has size and depth only a constant factor larger. The arithmetic circuit is also constant-free, and the because the depth of the initial Boolean circuit was $O(\log n)$, the degree of the polynomial that results is polynomial in n. Thus, if a language is in FNC^1, we can in logspace construct the corresponding constant-free polynomial-degree polynomial-size arithmetic circuit, so we're done. $\square$

Lemma 22.

$$\mathsf{GapP} \subseteq \mathsf{BP}(\mathsf{unif}_\mathsf{L}\text{-}\mathsf{VNP})$$

Proof. Follows immediately from Valiant's Criterion, Lemma 15, as for any $\phi \in \mathsf{GapP}$ we can take the polynomial

$$\sum_e \phi(e) \prod_i X_i^{e_i}$$

$\square$

Lemma 23.

$$\mathsf{BP}(\mathsf{unif}_{\mathsf{L}}\text{-}\mathsf{VNP}_{\mathbb{F}}) \subseteq \mathsf{GapP}$$

Proof. Suppose $\{P_n\} \in \mathsf{unif}_{\mathsf{L}}\text{-}\mathsf{VNP}_{\mathbb{F}}$. Then, per Lemma lm:valcrit-l 15, we can write it as

$$P_n = \sum_{c \in \{0,1\}^{n'}} \phi(c) \prod_{i=1}^{n} X_i^{c_i}$$

where c sums over binary representations of lists of integers, again following the representation and notation of Remark rm:c-notation 1

Now, because we're taking the Boolean part, we wish to evaluate this on an $e \in \{0,1\}^n$. The monomial terms that survive upon plugging in e are those containing only those variables X_i for which e_i is not 0. Define the support of c to be the set of indices

$$\operatorname{supp}(c) := \{i \in [n] \colon c_i \neq 0\},$$

so indices i where the variable X_i appears in the monomial. Recall that c_i refers to the number represented by the i^{th} block of c'. The support of e is similarly defined to be

$$\operatorname{supp}(e) := \{i \in [n] \colon e_i \neq 0\}$$

Formalizing what we said earlier,

$$\begin{aligned} P_n(e) &= \sum_{\substack{c \in \{0,1\}^{n'} \\ \operatorname{supp}(c) \subseteq \operatorname{supp}(e)}} \phi(c) \\ &= \sum_{\substack{c \in \{0,1\}^{n'} \\ \operatorname{supp}(c) \subseteq \operatorname{supp}(e)}} \left(\sum_{y \in \{0,1\}^m} f(c,y) - \sum_{y' \in \{0,1\}^{m'}} f'(c,y) \right) \end{aligned}$$

for some polynomially bounded m, m' and polynomial verifiers f and f'. But then,

$$P_n(e) = \sum_{(c,y) \in \{0,1\}^{u(n)}} f(c,y) \prod_{\substack{i \leq n \\ j \leq \deg(P_n)}} \Big(1 - c_{ij}(1-e_i)\Big) - \sum_{(c,y') \in \{0,1\}^{u'(n)}} f'(c,y) \prod_{\substack{i \leq n \\ j \leq \deg(P_n)}} \Big(1 - c_{ij}(1-e_i)\Big)$$

where $u(n) = m(n) + n'$ and $u'(n) = m'(n) + n'$. This exactly captures the behavior we want! The terms at the end ensure that in any surviving term, if $c_{ij} = 0$ for any j (equivalently, if $c_i \neq 0$), then e_i cannot be 0. To show that $P_n(e) \in \mathsf{GapP}$, it suffices to check that

$$g(E, C, Y) = f(C, Y) \prod_{\substack{i \leq n \\ j \leq \deg(P_n)}} \Big(1 - C_{ij}(1 - E_i)\Big)$$

is a polynomial time verifier, but this is clear because $f(C, Y)$ can be computed in polynomial time and there are polynomially many terms in the product, each of which is small. ☐

This seems to suggest, if we can show the single remaining inclusion (discussed in the next section, Conjecture 24 conj:l-unifvp-fnc3), that separating $\mathsf{unif_L\text{-}VP}$) from $\mathsf{unif_L\text{-}VNP}$) is as hard as separating $\#\mathsf{P}$ from NC.

Proving the Last Inclusion

conj:l-unifvp-fnc3 **Conjecture 24.**

$$\mathsf{BP}(\mathsf{unif_L\text{-}VP}_{\mathbb{F}}) \subseteq \mathsf{FNC}^3$$

Prelemmanaries

We'll use the definition of arithmetic circuit from the main document. All throughout, the polynomial computed at gate $\Gamma_\rho = (\omega_\rho \colon i_\rho, j_\rho)$ of an arithmetic circuit is b_ρ.

Definition 25. An arithmetic circuit Γ is **homogeneous** if all the intermediate results b_i are homogeneous. We say that f is computed by a homogeneous arithmetic circuit Γ if Γ computes all the homogeneous parts of f. $H(f)$ denotes the minimum size of a homogeneous arithmetic circuit computing the homogeneous parts of f

lm:homogenization **Lemma 26** ([Bür00] burgisser_algebraic-complexity-theory)**.** *For polynomials f of degree d, we have*

$$H(f) \leq (d+1)^2 L(f)$$

Proof. Let b_ρ^δ be the homogeneous part of the polynomial b_ρ of degree δ. Then, if $\omega_\rho = +$,

$$b_\rho^\delta = b_{i_\rho}^\delta + b_{j_\rho}^\delta$$

And if $\omega_\rho = \times$,

$$b_\rho^\delta = \sum_{0 \le k \le \delta} b_{i_\rho}^k b_{j_\rho}^{k-\delta}$$

Thus, it's sufficient to replace each gate b_ρ with $d+1$ gates, one for each degree part. Then, we just need extra gates to do the extra addition and multiplications. All in all, this comes to less than $(d+1)^2$ gates for each original gate of the circuit, so we're done. □

burgisser_algebraic-complexity-theory

lm:flattening **Lemma 27** ([Bür00])**.** *Let f be an n-variate polynomial of degree d. Then f can be computed by a homogeneous arithmetic circuit of size $O\left(d^6 L(f)^3\right)$ and depth $O\left(\log(dL(f)) \log d + \log n\right)$*

Proof. We won't prove correctness here. An outline of the algorithm is as follows (this will be useful downstream). Start with the initial size $L(f)$ arithmetic circuit for f, Γ. Define the following set:

This is a really poor sketch

$$\Gamma_b(a) := \{t \in r \colon t \in M, \deg(b_{i_t}), \deg(b_{j_t}) \le a < \deg(b_t)\}$$

where M is the set of multiplication gates that aren't multiplication by a scalar. Thus, $\Gamma_b(a)$ contains multiplication gates whose inputs have degree at most a but whose sum exceeds a

In the construction, we'll use polynomials b_{ij}, which we won't completely define here. We claim that the following are true for $\deg(b_j) \ge 2$

$$b_j = \sum_{t \in \Gamma_b(2^{\delta(j)})} b_t b_{tj} \tag{1}$$ eq:1

$$b_{ij} = \sum_{t \in \Gamma_b(2^{\delta(j)})} b_{it} b_{tj} \tag{2}$$ eq:2

where $2^{\delta(j)} < \deg(b_j) \le 2^{\delta(j)+1}$

Then, the circuit computed thusly has the desired size and depth.

1. Homogenize f, per Lemma lm:homogenization 26

2. Precompute those b_j and b_{ij} of degree 1 (there's a simple rule for computing the degree 1 b_{ij})

3. Compute every other b_j and b_{ij} using Equations eq:1 1 and eq:2 2.

□

lm:boolvpv-fnc3 **Lemma 28** ([Bür00]). *Over k a characteristic 0 field, and assuming GRH*

$$\mathsf{BP}(\mathsf{VP}_k) \subseteq \mathsf{FNC}^3/\mathsf{poly}$$

Proof. Let $f = \{f_n\}$ be a family of polynomials in $\mathsf{BP}(\mathsf{VP}_k)$. By Lemma 26 and Lemma 27, there exists a polynomial size arithmetic circuit of depth $O(\log^2 n)$ for computing f. Observe that $\deg f_n \leq 2^{O(\log^2 n)}$, and by Lemma 6, so too is $\log \operatorname{wt} f_n$. Replace the constants $\alpha = \alpha_1, \ldots, \alpha_m \in \mathbb{C}^m$ in f with the variables $Y = Y_1, \ldots, Y_m$, such that $F_n(X, Y) := F_n(X, \alpha) = f_n(x)$. Then, consider the system of equations

$$F_n(x, Y) = f(x) \forall x \in \{0,1\}^n$$

Then, per Lemma **??**, because we know this system is solvable over $\mathbb{C}$, it has a solution over a prime of size $2^{n^{O(1)}}$. We can replace each computation over $\mathbb{F}_p$ with a small circuit of size $O(\log p) = n^{O(1)}$ and depth $O(\log\log p) = O(\log n)$, thus increasing the depth by a factor of $\log n$ for a total depth of $O(\log^3 n)$, as desired. $\square$

Most of a Proof of Conjecture 24

lm:l-degree **Lemma 29.** *Consider an n-variate polynomial family $\{f_n\}$. Suppose there exists a logspace algorithm $\mathcal{A}$ that, on input 1^n, outputs arithmetic circuits of formal degree d computing the polynomial f_n, where $d = n^{O(1)}$. Then, there exists a logspace algorithm $\mathcal{D}$ that, on input 1^n, outputs $\tilde{d}$ such that $d \leq \tilde{d}$ and $\tilde{d} = n^{O(1)}$*

Prove this! It's important and seems non-trivial. More on this later

lm:l-homogenization **Lemma 30.** *Consider an n-variate polynomial family $\{f_n\}$. Suppose there exists a logspace algorithm $\mathcal{A}$ that, on input 1^n, outputs arithmetic circuits of formal degree d computing the polynomial f_n, where $d = n^{O(1)}$. Then, there exists a logspace algorithm $\mathcal{B}$ that, on input 1^n, outputs a homogeneous arithmetic circuit computing the homogeneous parts of f*

Proof of 30. First, by Lemma 29, we know there's an algorithm $\mathcal{D}$ that gets us an upper bound $\tilde{d}$ on d that still grows polynomially in n. Once we have an upper bound for the degree, we replace each gate with $(d+1)^2$ gates just as in Lemma 26.

$$b_\rho^\delta = b_{i_\rho}^\delta + b_{j_\rho}^\delta$$

$$b_\rho^\delta = \sum_{0 \leq k \leq \delta} b_{i_\rho}^k b_{j_\rho}^{k-\delta}$$

The algorithm is more formally written out in Algorithm 1.

Algorithm 1: Logspace Online Homogenization

```
// Get an upper bound on the degree of the circuit
$d \longleftarrow \mathcal{D}(1^n)$ // Run $\mathcal{A}$ and process each gate it outputs
for $(\omega : i, j) \in \mathcal{A}(1^n)$ do
    if $\omega ==' +'$ then
        for $0 \leq \delta \leq d$ do
            for $0 \leq k < B - (d+1)$ do
                // Output gates with value 0 until $B_3$
                output: $(+: 0, 0)$
            // The last $d+1$ bits of block $m$ store the
               homogeneous parts of gate $m$ in increasing order
            output: $(+: (i+1)B - (d+1) + \delta, (j+1)B - (d+1) + \delta)$
    if $\omega ==' \times'$ then
        // Output $(d+1)$ blocks of multiplication gates
        for $\delta \in [0, d]$ do
            // Output the multiplication gates for each $B_1$-sized
               block
            for $k \in [0, \delta]$ do
                output: $(\times: iB + \delta - k, jB + k)$
            // Fill out the rest of $B_1$ with 0s
            for $k \in [\delta + 1, B_1 - 1]$ do
                output: $(+: 0, (-1)0)$
        // Output $(d+1)$ blocks of addition gates
        for $\delta \in [0, d]$ do
            output: $(+: \rho B + \delta B_1, \rho B + \delta B_1 + 1)$
            for $k \in [0, d-3]$ do
                output: $(+: \rho B + (d+1)B_1 + \delta B_2 + k, \rho B + \delta B_1 + k + 2)$
        // Output $(d+1)$ gates, one per homogeneous part
        for $0 \leq \delta \leq d$ do
            output: $(+: \rho B + (d+1)B_1 + \delta B_2 + d - 2, \rho B + \delta B_1 + d - 1)$
```

Note that we assume, as per Definition 1, that we receive gates in topologically sorted order, starting with the variables, and must output them in the same way. Let's assign a block of B (this will end up being $O((\tilde{d}+1)^2)$ output arithmetic gates for each initial arithmetic gate Γ_ρ. We'll divide B into $(\tilde{d}+1)$ blocks of size B_1 (for computing pairwise products for each degree, as above), $(\tilde{d}+1)$ blocks of size B_2 (for adding up the pairwise products, and one block of size $B_3 = \tilde{d}+1$, which will be the actual output gates, one for each homogeneous part Γ_ρ^δ

□

lm:l-flattening **Lemma 31.** *Consider an n-variate polynomial family $\{f_n\}$. Suppose there exists a logspace algorithm $\mathcal{A}$ that, on input 1^n, outputs arithmetic circuits of formal degree d computing the polynomial f_n, where $d = n^{O(1)}$. Then, there's a logspace algorithm that, on input 1^n, outputs a constant-free homogeneous arithmetic circuit computing f of size $O\left(d^6 s^3\right)$ and depth $O(\log(ds)\log d + \log n)$*

lm:l-flattening lm:flattening

Proof of 31. Per the algorithm in Lemma 27, we'll start by outputting the gates corresponding to the degree 1 polynomials. We know which gates compute degree polynomials by the construction of the homogenization procedure, and so we know the IDs of such gates. For each, we can go back and look at the corresponding gate in the initial circuit, and output the appropriate linear combination of variables based on the original gate.

fill in more details and be more precise

Once we've done this, the remainder is simply outputting gates corresponding to the rules

$$b_j = \sum_{t \in \Gamma_b(2^{\delta(j)})} b_t b_{tj} \tag{3}$$

$$b_{ij} = \sum_{t \in \Gamma_b(2^{\delta(j)})} b_{it} b_{tj} \tag{4}$$

It's sufficient to iterate through the list of gates and values of i, and for each, go through all gates (all candidate t) and see if they are in $\Gamma_b(2^{\delta(j)}$. Given that the location of a gate is computable from its ID, so too is its degree. Thus, we can go through the list of t, and for each, output the Boolean logic corresponding to the product $b_{it} b_{tj}$. □

conj:l-unifvp-fnc3 lm:boolvpv-fnc3

Proof of Conjecture 24. We'll uniformize the approach in Lemma 28, and use it to construct the corresponding Boolean circuit gate-by-gate. The only part of the previous proof that wasn't plausibly constructive was the use of GRH to eliminate constants, but as it turns out, we won't need to - because we start with a constant-free arithmetic circuit (per Definition **??**), and because flattening the circuit as in Lemma 31 doesn't introduce any constants, the only constants in the flattened circuit are ± 1. Because every gate computes an integer polynomial, and we know the circuit computes a polynomial with a Boolean part, we can take the entire circuit "mod 2" and get the same result at the end. We can replace arithmetic gates with Boolean gates with an $O(1)$ increase in size and depth, and output said Boolean gates.

fill in more details and be more precise

unifVp lm:l-flattening

The main point of interest is that we can, before we even start outputting gates, know how many Boolean gates we'll output and which arithmetic gates they correspond to. Thus, we can output the final Boolean circuit gate-by-gate in any order (i.e. not necessarily in order of connectivity). We are able to do this because we have a fixed and systematic way of going from arithmetic to Boolean circuit. Our approach is as follows

1. Use a logspace approximation algorithm, as per Lemma **??** l-degree-approximation to obtain an upper bound $\tilde{d}$ for the degree. An approximation is good enough, as long as $\tilde{d}$ is still $n^{O(1)}$
2. Homogenize, as per Lemma **??** l-homogenization, blowing up each arithmetic circuit into $s(\tilde{d}+1)^2$ arithmetic gates.
3. Flatten, as per Lemma **??** l-flattening, which adds roughly $s^2(\tilde{d}+1)^4$ gates (because we need to compute the b_{ij}) write out what exactly the roughly is
4. Lastly, replace each of the resulting arithmetic gates with $O(1)$ Boolean gates.

Because we can know s and $\tilde{d}$ before we actually start outputting Boolean gates, we can know the set of Boolean gates (i.e. the addresses) corresponding to any arithmetic gate. Thus, while the steps above were written in sequential order, they actually operate as nested for loops. □

The Last Part of a Proof of Conjecture 24 conj:l-unifvp-fnc3

The only remaining step is getting a decent upper bound on the degree in logspace.

lm:multcircuit-degree **Lemma 32.** *The degree of an arithmetic circuit with only multiplication gates is equal to the number of paths from the output node s to an input variable*

Proof. Follows immediately from induction, on the size of the circuit - the degree at the topmost multiplication gate is the sum of the degrees of its children. The number of paths from the output node s to the input variables is equal to the number of paths starting at its left child plus the number of paths starting at its right child. □

Lemma 33. *Computing the degree of a homogeneous arithmetic circuit of size s reduces to computing the determinant of an $2s \times 2s$ matrix*

Proof. Consider walks from the output node s down to the inputs (so in the opposite direction from the usual arrows of the circuit), and then add edges from all input variables (not constants, just variables) to a sink node t. When computing the degree, $+$ gates don't matter. Thus, in walks from s to t, we ignore $+$ gates by always going to the right child. Once we've established this, we can treat our circuit as if it only has $\times$ gates and apply Lemma 32 to see that the degree is the number of paths connnecting s and t. To turn this into a determinant, start with the adjacency matrix of the circuit (where gates are nodes and wires are edges). Add a node in the middle of each edge (so every path between two nodes in the original graph is now of even length), add self-loops to all $2s$ nodes, and add an edge between s and t. Because the original graph was acyclic, any cycle cover of this graph consists of one "large" cycle containing s, t, and the special edge between them, and self-loops for every node not in the large cycle. Note that the large cycle has odd length because the path from s to t is of even length. Thus, the sign of every cycle cover is 1. Therefore, the determinant of this adjacency matrix is exactly the sum of products of the weights of its cycle covers. If we assign a weight of 1 to every edge, then the determinant counts the number of "large cycles" including s and t, or equivalently, the number of paths from s to t. □

Corollary 34. *The degree of a homogeneous arithmetic circuit of size s can be computed in* NC^2

Proof. Per [Ber84], computing the determinant is in NC^2 □

References

arora_computational-complexity Arora, Sanjeev and Boaz Barak. *Computational Complexity: A Modern Approach.* 1st. New York, NY, USA: Cambridge University Press, 2009. ISBN: 0521424267, 9780521424264.

berkowitz_computing_1984 Berkowitz, Stuart J. "On computing the determinant in small parallel time using a small number of processors". In: *Information Processing Letters* 18.3 (Mar. 30, 1984), pp. 147–150. ISSN: 0020-0190. DOI: `10.1016/0020-0190(84)90018-8`. URL: `http://www.sciencedirect.com/science/article/pii/0020019084900188` (visited on 08/08/2018).

burgisser_algebraic-complexity-theory Bürgisser, Peter. *Completeness and Reduction in Algebraic Complexity Theory.* 1st ed. Vol. 7. Algorithms and Computation in Mathematics. Springer-Verlag Berlin Heidelberg, 2000. ISBN: 3-540-66752-0.

fournier_fixed-polynomial_2013 Fournier, Hervé, Sylvain Perifel, and Rémi de Verclos. "On Fixed-Polynomial Size Circuit Lower Bounds for Uniform Polynomials in the Sense of Valiant". In: *Mathematical Foundations of Computer Science 2013*. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Aug. 26, 2013, pp. 433–444. ISBN: 978-3-642-40312-5 978-3-642-40313-2. DOI: `10.1007/978-3-642-40313-2_39`. URL: `https://link.springer.com/chapter/10.1007/978-3-642-40313-2_39` (visited on 06/12/2018).

koiran_hilberts_1996 Koiran, Pascal. "Hilbert's Nullstellensatz Is in the Polynomial Hierarchy". In: *J. Complex.* 12.4 (Dec. 1996), pp. 273–286. ISSN: 0885-064X. DOI: `10.1006/jcom.1996.0019`. URL: `http://dx.doi.org/10.1006/jcom.1996.0019` (visited on 07/19/2018).

vinodchandran_pp-circuit-complexity V. Vinodchandran, N. "A note on the circuit complexity of PP." In: 347 (Jan. 2005), pp. 415–418.