Extending CC0 Circuit Upper Bounds Beyond Symmetric Functions

Luv Udeshi

PRIMES-USA

October 2025

Mentor: Brynmor Chapman

Outline

Motivation

Definition

Consider a Boolean function f. We call the set $\{x : f(x) = 1\}$ a language/decision problem associated with f.

Motivation

Definition

Consider a Boolean function f. We call the set $\{x : f(x) = 1\}$ a language/decision problem associated with f.

Definition

A complexity class is a set of languages.

Motivation

Definition

Consider a Boolean function f. We call the set $\{x : f(x) = 1\}$ a language/decision problem associated with f.

Definition

A complexity class is a set of languages.

- It is of interest to us to look at complexity classes of efficiently computable languages, i.e. sets of languages that can be computed within a certain resource constraint.

Motivation

Definition

Consider a Boolean function f. We call the set $\{x : f(x) = 1\}$ a language/decision problem associated with f.

Definition

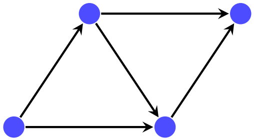
A complexity class is a set of languages.

- It is of interest to us to look at complexity classes of efficiently computable languages, i.e. sets of languages that can be computed within a certain resource constraint.
- Circuits are mathematically simpler and are much more powerful than classical computation models, like Turing Machines.

Graphs

Definition (Directed Graph)

A *directed graph* is an ordered pair $G = (V, E)$, where V is a finite set of vertices, and $E \subseteq V \times V$ is a set of ordered pairs (u, v) called directed edges, where each edge goes from vertex u to vertex v.



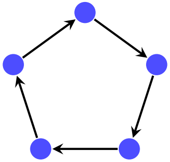
Graphs

Definition (Cycle in a Directed Graph)

A *cycle* in a directed graph is a sequence of distinct vertices

$$v_1, v_2, \ldots, v_k \quad (k \geq 2)$$

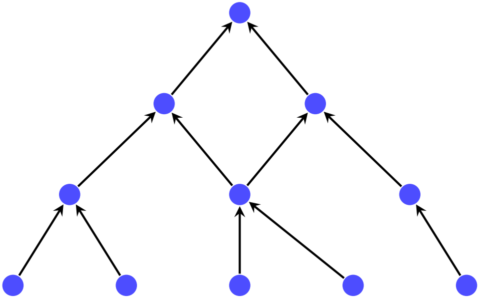
such that $(v_i, v_{i+1}) \in E$ for all $1 \leq i < k$, and additionally $(v_k, v_1) \in E$.



Graphs

Definition (Directed Acyclic Graph)

A *directed acyclic graph* is a directed graph that contains no directed cycles.



What is a circuit?

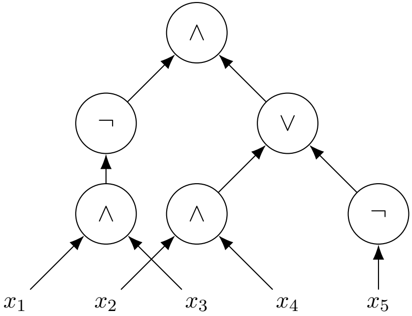
Definition

A *circuit* is a directed acyclic graph in which nodes of in-degree 0 are called *inputs* and all other nodes are called *gates*. Gates of out-degree 0 are called *outputs*. Each gate g of in-degree k is labeled with a k-ary function f_g, and computes it in a natural way.

What is a circuit?

Definition

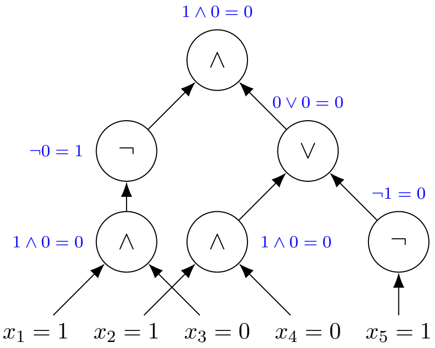
A *circuit* is a directed acyclic graph in which nodes of in-degree 0 are called *inputs* and all other nodes are called *gates*. Gates of out-degree 0 are called *outputs*. Each gate g of in-degree k is labeled with a k-ary function f_g, and computes it in a natural way.



What is a circuit?

Definition

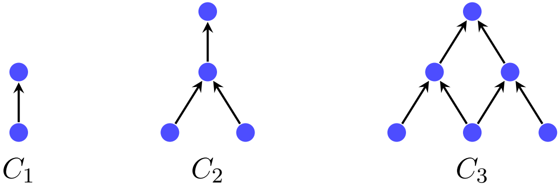
A *circuit* is a directed acyclic graph in which nodes of in-degree 0 are called *inputs* and all other nodes are called *gates*. Gates of out-degree 0 are called *outputs*. Each gate g of in-degree k is labeled with a k-ary function f_g, and computes it in a natural way.



Circuit families

Definition

A *circuit family* is a sequence $\{C_n\}_{n\in\mathbb{N}}$ of circuits, where C_n has n inputs and a single output.



Circuit families

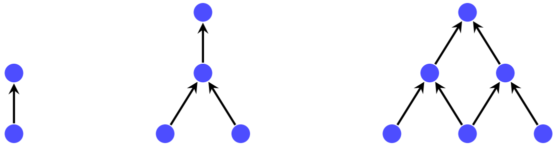
- The size of a circuit is the total number of gates it contains.

Circuit families

- The size of a circuit is the total number of gates it contains.

Definition

For a function $T : \mathbb{N} \to \mathbb{N}$ such that $T(n) \geq |C_n|$ for all $n \in \mathbb{N}$, we say $\{C_n\}$ has size $T(n)$.



C_1 $|C_1| = 1$

C_2 $|C_2| = 2$

C_3 $|C_3| = 3$

Circuit families

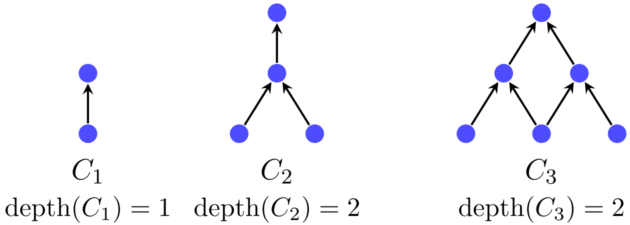
- The depth of a circuit is the length of the longest path from an input to an output.

Circuit families

- The depth of a circuit is the length of the longest path from an input to an output.

Definition

For a function $T : \mathbb{N} \to \mathbb{N}$ such that $T(n) \geq \mathrm{depth}(C_n)$ for all $n \in \mathbb{N}$, we say $\{C_n\}$ has depth $T(n)$.



Outline

Relevant Boolean functions

- The n-ary Boolean function $\mathrm{AND} : \{0,1\}^n \to \{0,1\}$ outputs 1 iff all of its inputs are 1.

Relevant Boolean functions

- The n-ary Boolean function $\mathrm{AND} : \{0,1\}^n \to \{0,1\}$ outputs 1 iff all of its inputs are 1.
- The n-ary Boolean function $\mathrm{OR} : \{0,1\}^n \to \{0,1\}$ outputs 1 iff at least one of its inputs is 1.

Relevant Boolean functions

- The n-ary Boolean function $\text{AND} : \{0,1\}^n \to \{0,1\}$ outputs 1 iff all of its inputs are 1.
- The n-ary Boolean function $\text{OR} : \{0,1\}^n \to \{0,1\}$ outputs 1 iff at least one of its inputs is 1.
- For a fixed positive integer m, the n-ary function $\text{MOD}_m : \{0,1\}^n \to \{0,1\}$ outputs 1 iff the sum of its inputs is divisible by m.

Relevant Boolean functions

- The n-ary Boolean function $\mathrm{AND} : \{0,1\}^n \to \{0,1\}$ outputs 1 iff all of its inputs are 1.
- The n-ary Boolean function $\mathrm{OR} : \{0,1\}^n \to \{0,1\}$ outputs 1 iff at least one of its inputs is 1.
- For a fixed positive integer m, the n-ary function $\mathrm{MOD}_m : \{0,1\}^n \to \{0,1\}$ outputs 1 iff the sum of its inputs is divisible by m.
- The three functions above are examples of symmetric functions.

Relevant Boolean functions

- The n-ary Boolean function $\mathrm{AND} : \{0,1\}^n \to \{0,1\}$ outputs 1 iff all of its inputs are 1.
- The n-ary Boolean function $\mathrm{OR} : \{0,1\}^n \to \{0,1\}$ outputs 1 iff at least one of its inputs is 1.
- For a fixed positive integer m, the n-ary function $\mathrm{MOD}_m : \{0,1\}^n \to \{0,1\}$ outputs 1 iff the sum of its inputs is divisible by m.
- The three functions above are examples of symmetric functions.

Definition

A Boolean function f is called *symmetric* if there exists $g : \mathbb{N} \to \{0,1\}$ such that for each binary vector $\mathbf{x}$,

$$f(\mathbf{x}) = g\left(|\mathbf{x}|_1\right),$$

i.e., it depends only on the number of 1s in the input.

Circuit Complexity Classes

Definition

An *alternating circuit* is a circuit in which every gate computes an AND or OR function of unbounded in-degree.

Circuit Complexity Classes

Definition

An *alternating circuit* is a circuit in which every gate computes an AND or OR function of unbounded in-degree.

- AC^i is the class of languages computable by alternating circuit families of polynomial size and depth $O(\log^i n)$.

Circuit Complexity Classes

Definition

An *alternating circuit* is a circuit in which every gate computes an AND or OR function of unbounded in-degree.

- AC^i is the class of languages computable by alternating circuit families of polynomial size and depth $O(\log^i n)$.

Definition

A *counting circuit* is a circuit in which every gate computes a MOD function of unbounded in-degree.

Circuit Complexity Classes

Definition

An *alternating circuit* is a circuit in which every gate computes an AND or OR function of unbounded in-degree.

- AC^i is the class of languages computable by alternating circuit families of polynomial size and depth $O(\log^i n)$.

Definition

A *counting circuit* is a circuit in which every gate computes a MOD function of unbounded in-degree.

- $\mathsf{CC}^i[m]$ is the class of languages computable by counting circuit families of polynomial size and depth $O(\log^i n)$, where every gate computes the MOD_m function.

Circuit Complexity Classes (cont.)

Definition

An *alternating circuit with counting* is a circuit in which every gate computes either a MOD function, the AND function, or the OR function, all of unbounded in-degree.

Circuit Complexity Classes (cont.)

Definition

An *alternating circuit with counting* is a circuit in which every gate computes either a MOD function, the AND function, or the OR function, all of unbounded in-degree.

- AC$^i[m]$ is the class of languages computable by alternating counting circuit families of polynomial size and depth $O(\log^i n)$, where every MOD gate computes the MOD_m function.

Circuit Complexity Classes (cont.)

Definition

An *alternating circuit with counting* is a circuit in which every gate computes either a MOD function, the AND function, or the OR function, all of unbounded in-degree.

- $\mathsf{AC}^i[m]$ is the class of languages computable by alternating counting circuit families of polynomial size and depth $O(\log^i n)$, where every MOD gate computes the MOD_m function.
- We will focus on CC^0.

Outline

Key lower bound

Theorem (Razborov, Smolensky)

For distinct primes $p \neq q$, any depth-d $\mathrm{AC}^0[q]$ circuit that computes MOD_p must have size at least $\exp\big(\Omega(n^{1/(2d)})\big)$.

Key lower bound

Theorem (Razborov, Smolensky)

For distinct primes $p \neq q$*, any depth-*d $\mathrm{AC}^0[q]$ *circuit that computes* MOD_p *must have size at least* $\exp\big(\Omega(n^{1/(2d)})\big)$*.*

- This is a powerful lower bound which shows the weakness of prime-modulus counting circuits.

Key lower bound

Theorem (Razborov, Smolensky)

For distinct primes $p \neq q$, any depth-d $\mathrm{AC}^0[q]$ circuit that computes MOD_p must have size at least $\exp\left(\Omega(n^{1/(2d)})\right)$.

- This is a powerful lower bound which shows the weakness of prime-modulus counting circuits.
- Motivates us to consider composite moduli.

Outline

Symmetric Upper Bounds

Theorem (Chapman, Williams)

For every $\varepsilon > 0$, there is a modulus $m \leq (1/\varepsilon)^{2/\varepsilon}$ such that every symmetric function on n bits can be computed by depth-3 MOD_m circuits of $\exp(O(n^\varepsilon))$ size.

Symmetric Upper Bounds

Theorem (Chapman, Williams)

For every $\varepsilon > 0$, there is a modulus $m \leq (1/\varepsilon)^{2/\varepsilon}$ such that every symmetric function on n bits can be computed by depth-3 MOD_m circuits of $\exp(O(n^{\varepsilon}))$ size.

- In particular, the construction yields circuits of size $\exp(O(n^{1/k} \log n))$ where $m = p_1 \ldots p_k$ is a product of k distinct primes.

Symmetric Upper Bounds

Theorem (Chapman, Williams)

For every $\varepsilon > 0$*, there is a modulus* $m \leq (1/\varepsilon)^{2/\varepsilon}$ *such that every symmetric function on* n *bits can be computed by depth-*3 MOD_m *circuits of* $\exp(O(n^{\varepsilon}))$ *size.*

- In particular, the construction yields circuits of size $\exp(O(n^{1/k}\log n))$ where $m = p_1 \ldots p_k$ is a product of k distinct primes.

Example

Every symmetric Boolean function on n variables has depth-three MOD_{30} circuits of size $\exp(O(n^{1/3}\log n))$.

Outline

Generalization

- Symmetric functions are those that depend only on the polynomial $\sum_{i=1}^{n} x_i$. If we have a function that is instead dependent only on the value of a different polynomial, we can compute it in a similar manner.

Generalization

- Symmetric functions are those that depend only on the polynomial $\sum_{i=1}^{n} x_i$. If we have a function that is instead dependent only on the value of a different polynomial, we can compute it in a similar manner.

Theorem

Let f be a Boolean function on n bits $x_1, \ldots, x_n$ and m be a product of $k \geq 2$ distinct primes. Let $I(x) \in \mathbb{Z}[x_1, \ldots, x_n]$ of degree d, whose maximum value over all Boolean inputs is M. Then, every f that depends solely on the value of $I(x)$ can be computed by depth-3 MOD_m circuits of size $\exp(O(d \cdot M^{1/k} \log M))$.

Example

Example

Consider $f(x) = \bigvee_{i=1}^{n} (x_i \wedge x_{i+1})$.

Example

Example

Consider $f(x) = \bigvee_{i=1}^{n} (x_i \wedge x_{i+1})$.

- What is this function doing? How can we mimic this with a simple polynomial?

Example

Example

Consider $f(x) = \bigvee_{i=1}^{n} (x_i \wedge x_{i+1})$.

- What is this function doing? How can we mimic this with a simple polynomial?
- We can use the degree-two polynomial $I(x) = \sum_{i=1}^{n} x_i x_{i+1}$ to give us depth-three MOD_m circuits of size $\exp(O(n^{1/k} \log n))$ computing f.

Symmetry Composition

Theorem

Let $m = p_1 \cdots p_k$ be a product of k distinct primes, and let g be an n-variate symmetric function. For each $1 \leq i \leq n$, let h_i be a function that depends only on n^ϵ inputs. Then $f(\mathbf{x}) := g(h_1(\mathbf{x}), \ldots, h_n(\mathbf{x}))$ can be computed with depth-three MOD_m circuits of size $\exp(O(n^{2/k+\epsilon} \log m))$.

Symmetry Composition

Theorem

Let $m = p_1 \cdots p_k$ be a product of k distinct primes, and let g be an n-variate symmetric function. For each $1 \le i \le n$, let h_i be a function that depends only on n^ϵ inputs. Then $f(\mathbf{x}) := g(h_1(\mathbf{x}), \ldots, h_n(\mathbf{x}))$ can be computed with depth-three MOD_m circuits of size $\exp(O(n^{2/k+\epsilon} \log m))$.

Corollary

Let g be an n-variate symmetric function. For each $1 \le i \le n$, let h_i be a function that depends only on a subpolynomial number of inputs. Then $f(\mathbf{x}) := g(h_1(\mathbf{x}), \ldots, h_n(\mathbf{x}))$ can be computed with a counting circuit family of depth three and subexponential size.

Acknowledgements

- I kindly thank my PRIMES mentor, Brynmor Chapman, for proposing this project and for his guidance. I would also like to express gratitude to PRIMES for making this research experience possible.

Outline

References

- S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.
- David A. Mix Barrington, Richard Beigel, and Steven Rudich. Representing Boolean functions as polynomials modulo composite numbers. *Comput. Complexity*, 4:367–382, 1994.
- Brynmor Chapman and Ryan Williams. Smaller ACC0 circuits for symmetric functions. *arXiv preprint arXiv:2107.04706*, 2021.
- Alexander A. Razborov. Lower bounds on the size of bounded-depth networks over the complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.
- Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In STOC, pages 77–82, 1987.