Constant-Depth Circuits vs. Monotone Circuits

Bruno P. Cavalar∗
Department of Computer Science
University of Warwick

Igor C. Oliveira†
Department of Computer Science
University of Warwick

May 11, 2023

Abstract

We establish new separations between the power of monotone and general (non-monotone) Boolean circuits:

– For every $k \geq 1$, there is a monotone function in $\mathsf{AC}^0$ (constant-depth poly-size circuits) that requires monotone circuits of depth $\Omega(\log^k n)$. This significantly extends a classical result of Okol’nishnikova [Oko82] and Ajtai and Gurevich [AG87]. In addition, our separation holds for a monotone graph property, which was unknown even in the context of $\mathsf{AC}^0$ versus $\mathsf{mAC}^0$.

– For every $k \geq 1$, there is a monotone function in $\mathsf{AC}^0[\oplus]$ (constant-depth poly-size circuits extended with parity gates) that requires monotone circuits of size $\exp(\Omega(\log^k n))$. This makes progress towards a question posed by Grigni and Sipser [GS92].

These results show that constant-depth circuits can be more efficient than monotone formulas and monotone circuits when computing monotone functions.

In the opposite direction, we observe that non-trivial simulations are possible in the absence of parity gates: every monotone function computed by an $\mathsf{AC}^0$ circuit of size $s$ and depth $d$ can be computed by a monotone circuit of size $2^{n-\log s/d-1}$. We show that the existence of significantly faster monotone simulations would lead to breakthrough circuit lower bounds. In particular, if every monotone function in $\mathsf{AC}^0$ admits a polynomial size monotone circuit, then $\mathsf{NC}^2$ is not contained in $\mathsf{NC}^1$.

Finally, we revisit our separation result against monotone circuit size and investigate the limits of our approach, which is based on a monotone lower bound for constraint satisfaction problems (CSPs) established by Gőös, Kamath, Robere and Sokolov [GKRS19] via lifting techniques. Adapting results of Schaefer [Sch78] and Allender, Bauland, Immerman, Schnoor and Vollmer [ABI+09], we obtain an unconditional classification of the monotone circuit complexity of Boolean-valued CSPs via their polymorphisms. This result and the consequences we derive from it might be of independent interest.

∗Email: bruno.pasqualotto-cavalar@warwick.ac.uk
†Email: igor.oliveira@warwick.ac.uk
# Contents

1 Introduction .......................................................... 3
   1.1 Results .................................................................. 4
       1.1.1 Constant-depth circuits vs. monotone circuits ..... 5
       1.1.2 Non-trivial monotone simulations and their consequences ..... 5
       1.1.3 Monotone complexity of constraint satisfaction problems .... 6
   1.2 Techniques ............................................................ 8
   1.3 Directions and open problems .................................... 11

2 Preliminaries .............................................................. 12
   2.1 Notation ............................................................... 12
   2.2 Background results ................................................ 13

3 Constant-Depth Circuits vs. Monotone Circuits ................. 13
   3.1 A monotone size lower bound for a function in $AC^0[⊕]$ .......... 13
   3.2 A monotone depth lower bound for a graph property in $AC^0$ ... 15
   3.3 Efficient monotone padding for graph properties .................. 17

4 Non-Trivial Monotone Simulations and Their Consequences ... 18
   4.1 A non-trivial simulation for bounded-depth circuits .............. 19
   4.2 Non-monotone lower bounds from monotone simulations ........ 20

5 Monotone Complexity of Constraint Satisfaction Problems .... 22
   5.1 Definitions .......................................................... 22
   5.2 Basic facts about CSP-SAT ....................................... 24
   5.3 A monotone dichotomy for CSP-SAT ............................ 24
   5.4 Some auxiliary results ............................................. 30
   5.5 Consequences for monotone circuit lower bounds via lifting .... 31

A A Lower Bound for 3-XOR-SAT Using the Approximation Method .... 37

B Schaefer’s Theorem in Monotone Complexity ..................... 39
   B.1 Connectivity and generation functions .......................... 39
   B.2 Proof of reduction lemmas ....................................... 39
   B.3 Monotone circuit upper bounds ................................. 40

C Background on Post’s Lattice and Clones .......................... 41
1 Introduction

A Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is monotone if $f(x) \leq f(y)$ whenever $x_i \leq y_i$ for each coordinate $1 \leq i \leq n$. Monotone Boolean functions, and the monotone Boolean circuits\footnote{Recall that in a monotone Boolean circuit the gate set is limited to \{AND, OR\} and input gates are labelled by elements from \{x_1, \ldots, x_n, 0, 1\}.} that compute them, have been extensively investigated for decades due to their relevance in circuit complexity [Raz85a], cryptography [BL88], learning theory [BT96], proof complexity [Kra97, Pud97], property testing [GGLR98], pseudorandomness [CZ16], optimisation [GJW18], hazard-free computations [IKL+19], and meta-complexity [Hir22], among other topics. In addition, over the last few years a number of results have further highlighted the importance of monotone complexity as a central topic in the study of propositional proofs, total search problems, communication protocols, and related areas (see [dRGR22] for a recent survey).

Some of the most fundamental results about monotone functions deal with their complexities with respect to different classes of Boolean circuits, such as the monotone circuit lower bound of Razborov [Raz85b] for Matching and the constant-depth circuit lower bound of Rossman [Ros08b] for $k$-Clique. Particularly important to our discussion is a related strand of research that contrasts the computational power of monotone circuits relative to general (non-monotone) AND/OR/NOT circuits, which we review next.

Weakness of Monotone Circuits. The study of monotone simulations of non-monotone computations and associated separation results has a long and rich history. In a sequence of celebrated results, [Raz85b, And85, AB87, Tar88] showed the existence of monotone functions that can be computed by circuits of polynomial size but require monotone circuits of size $2^{\Omega(n)}$. In other words, the use of negations can significantly speedup the computation of monotone functions. More recently, G"{o}"os, Kamath, Robere and Sokolov [GKRS19] considerably strengthened this separation by showing that some monotone functions in \text{NC}^2 (poly-size $O(\log^2 n)$-depth fan-in two circuits) require monotone circuits of size $2^{n^{\Omega(1)}}$. (An earlier weaker separation against monotone depth $n^{\Omega(1)}$ was established in [RW92].) Therefore, negations can also allow monotone functions to be efficiently computed in parallel.

Similar separations about the limitations of monotone circuits are also known at the low-complexity end of the spectrum: Okol’nishnikova [Oko82] and (independently) Ajtai and Gurevich [AG87] exhibited monotone functions in \text{AC}^0 (i.e., constant-depth poly-size AND/OR/NOT circuits) that require monotone \text{AC}^0 circuits (composed of only AND/OR gates) of super-polynomial size.\footnote{We refer to [BST13] for an alternate exposition of this result.} This result has been extended to an exponential separation in [COS17], which shows the existence of a monotone function in \text{AC}^0 that requires monotone depth-$d$ circuits of size $2^{\tilde{\Omega}(n^{1/d})}$ even if \text{MAJ} (majority) gates are allowed in addition to AND/OR gates.\footnote{Separations between monotone and non-monotone devices have also been extensively investigated in other settings. This includes average-case complexity [BHST14], different computational models, such as span programs [BGW99, RPRC16] and algebraic complexity (see [CDM21] and references therein), and separations in first-order logic [Sto95, Kup21, Kup22]. We restrict our attention to worst-case separations for Boolean circuits in this paper.}

Strength of Monotone Circuits. In contrast to these results, in many settings negations do not offer a significant speedup and monotone computations can be unexpectedly powerful. For
instance, monotone circuits are able to efficiently implement several non-trivial algorithms, such as solving constraint satisfaction problems using treewidth bounds (see, e.g., [Oli15, Chapter 3]). As another example, in the context of cryptography, it has been proved that if one-way functions exist, then there are monotone one-way functions [GI12]. Below we describe results that are more closely related to the separations investigated in our paper.

In the extremely constrained setting of depth-2 circuits, Quine [Qui53] showed that monotone functions computed by size-$s$ DNFs (resp., CNFs) can always be computed by size-$s$ monotone DNFs (resp., CNFs). Some results along this line are known for larger circuit depth, but with respect to more structured classes of monotone Boolean functions. Rossman [Ros08a, Ros17b] showed that any homomorphism-preserving graph property computed by $AC^0$ circuits is also computed by monotone $AC^0$ circuits.\footnote{A function $f : \{0, 1\}^n \to \{0, 1\}$ is called a graph property if $f(G) = f(H)$ whenever $G$ and $H$ are isomorphic graphs, and homomorphism-preserving if $f(G) \leq f(H)$ whenever there is a graph homomorphism from $G$ to $H$. It is easy to see that every homomorphism-preserving graph property is monotone.} Under no circuit depth restriction, Berkowitz [Ber82] proved that the monotone and non-monotone circuit size complexities of every slice function are polynomially related.\footnote{A function $f : \{0, 1\}^n \to \{0, 1\}$ is a slice function if there is $i \geq 0$ such that $f(x)$ is 0 on inputs of Hamming weight less than $i$ and 1 on inputs of Hamming weight larger than $i$.}

Despite much progress and sustained efforts, these two classes of results leave open tantalising problems about the power of cancellations in computation.\footnote{Any non-monotone circuit can be written as an XOR (parity) of distinct monotone sub-circuits (see, e.g., [GMOR15, Appendix A.1]), so negations can be seen as a way of combining, or cancelling, different monotone computations. See also a related discussion in Valiant [Val80].} In particular, they suggest the following basic question about the contrast between the weakness of monotone computations and the strength of negations:

**What is the largest computational gap between the power of monotone and general (non-monotone) Boolean circuits?**

A concrete formalisation of this question dates back to the seminal work on monotone complexity of Grigni and Sipser [GS92] in the early nineties. They asked if there are monotone functions in $AC^0$ that require super-polynomial size monotone Boolean circuits, i.e., if $AC^0 \cap \text{Mono} \not\subseteq m\text{SIZE}[\text{poly}]$. In case this separation holds, it would exhibit the largest qualitative gap between monotone and general Boolean circuits, i.e., even extremely parallel non-monotone computations can be more efficient than arbitrary monotone computations.

### 1.1 Results

Our results show that, with respect to the computation of monotone functions, highly parallel (non-monotone) Boolean circuits can be super-polynomially more efficient than unrestricted monotone circuits. Before providing a precise formulation of these results, we introduce some notation.

For a function $d : \mathbb{N} \to \mathbb{N}$, let $m\text{DEPTH}[d]$ denote the class of Boolean functions computed by monotone fan-in two AND/OR Boolean circuits of depth $O(d(n))$. Similarly, we use $m\text{SIZE}[s]$ to denote the class of Boolean functions computed by monotone circuits of size $O(s(n))$. More generally, for a circuit class $\mathcal{C}$, we let $m\mathcal{C}$ denote its natural monotone analogue. Finally, for a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$, we use $m\text{SIZE}(f)$ and $m\text{DEPTH}(f)$ to denote its monotone size and depth complexities, respectively.

Our results show that, with respect to the computation of monotone functions, highly parallel (non-monotone) Boolean circuits can be super-polynomially more efficient than unrestricted monotone circuits. Before providing a precise formulation of these results, we introduce some notation.

For a function $d : \mathbb{N} \to \mathbb{N}$, let $m\text{DEPTH}[d]$ denote the class of Boolean functions computed by monotone fan-in two AND/OR Boolean circuits of depth $O(d(n))$. Similarly, we use $m\text{SIZE}[s]$ to denote the class of Boolean functions computed by monotone circuits of size $O(s(n))$. More generally, for a circuit class $\mathcal{C}$, we let $m\mathcal{C}$ denote its natural monotone analogue. Finally, for a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$, we use $m\text{SIZE}(f)$ and $m\text{DEPTH}(f)$ to denote its monotone size and depth complexities, respectively.
circuit size and depth complexities, respectively. We refer to Jukna [Juk12] for standard background on circuit complexity theory.

1.1.1. Constant-depth circuits vs. monotone circuits. Recall that the Okol’nishnikova-Ajtai-Gurevich [Oko82, AG87] theorem states that $\text{AC}^0 \cap \text{Mono} \not\subseteq \text{mAC}^0$. In contrast, as our main result, we establish a separation between constant-depth Boolean circuits and monotone circuits of much larger depth. In particular, we show that constant-depth circuits with negations can be significantly more efficient than monotone formulas.

**Theorem 1.1** (Polynomial-size constant-depth vs. larger monotone depth). For every $k \geq 1$, we have $\text{AC}^0 \cap \text{Mono} \not\subseteq \text{mDEPTH}[\log(n)^k]$. Moreover, this separation holds for a monotone graph property.

In a more constrained setting, Kuperberg [Kup21, Kup22] exhibited a monotone graph property expressible in first-order logic that cannot be expressed in positive first-order logic. A separation that holds for a monotone graph property was unknown even in the context of $\text{AC}^0$ versus $\text{mAC}^0$.

Let $\text{HomPreserving}$ denote the class of all homomorphism-preserving graph properties, and recall that Rossman [Ros08a, Ros17b] established that $\text{AC}^0 \cap \text{HomPreserving} \subseteq \text{mAC}^0$. Theorem 1.1 implies that this efficient monotone simulation does not extend to the larger class of monotone graph properties, even if super-logarithmic depth is allowed.

Our argument is completely different from those of [Oko82, AG87, BST13, COS17] and their counterparts in first-order logic [Sto95, Kup21, Kup22]. In particular, it allows us to break the $O(\log n)$ monotone depth barrier present in previous separations with an $\text{AC}^0$ upper bound, which rely on lower bounds against monotone circuits of depth $d$ and size (at most) $2^{O(1/d)}$. We defer the discussion of our techniques to Section 1.2.

In our next result, we consider monotone circuits of unbounded depth.

**Theorem 1.2** (Polynomial-size constant-depth vs. larger monotone size). For every $k \geq 1$, we have $\text{AC}^0[\oplus] \cap \text{Mono} \not\subseteq \text{mSIZE}[2^{\log(n)^k}]$.

Theorem 1.1 and Theorem 1.2 are incomparable: while the monotone lower bound is stronger in the latter, its constant-depth upper bound requires parity gates. Theorem 1.2 provides the first separation between constant-depth circuits and monotone circuits of polynomial size, coming remarkably close to a solution to the question considered by Grigni and Sipser [GS92].

We note that in both of our results the family of monotone functions is explicit and has a simple description (see Section 1.2).

1.1.2. Non-trivial monotone simulations and their consequences. While Theorem 1.1 and Theorem 1.2 provide more evidence for the existence of monotone functions in $\text{AC}^0$ which require monotone circuits of super-polynomial size, they still leave open the intriguing possibility that unbounded fan-in $\oplus$-gates might be crucial to achieve the utmost cancellations (speedups) provided by constant-depth circuits. This further motivates the investigation of efficient monotone simulations of constant-depth circuits without parity gates, which we consider next.

For convenience, let $\text{AC}^0_{d}[s]$ denote the class of Boolean functions computed by $\text{AC}^0$ circuits of depth $\leq d$ and size $\leq s(n)$. (We might omit $s(n)$ and/or $d$ when implicitly quantifying over all families of polynomial size circuits and/or all constant depths.)
We observe that a non-trivial monotone simulation is possible in the absence of parity gates. Indeed, by combining existing results from circuit complexity theory, it is not hard to show that $AC^0_d[8] \cap \text{Mono} \subseteq \text{mSIZE}[2^{O(1-1/O(s^{d-1})]}]$ (see Section 4.1). Moreover, this upper bound is achieved by monotone DNFs of the same size. This is the best upper bound we can currently show for the class of all monotone functions when the depth $d \geq 3$. (Negations offer no speedup at depths $d \leq 2$ [Qui53].) In contrast, we prove that a significantly faster monotone simulation would lead to new (non-monotone) lower bounds in complexity theory. Recall that it is a notorious open problem to obtain explicit lower bounds against depth-$d$ circuits of size $2^{\omega(n^{1/(d-1)})}$, for any fixed $d \geq 3$. We denote by GraphProperties the set of all Boolean functions which are graph properties.

**Theorem 1.3** (New circuit lower bounds from monotone simulations). There exists $\varepsilon > 0$ such that the following holds.

1. If $AC^0_3 \cap \text{Mono} \subseteq \text{mNC}^1$, then $NP \not\subseteq AC^0_3[2^{o(n)}]$.

2. If $AC^0_4 \cap \text{Mono} \subseteq \text{mSIZE}[\text{poly}]$, then $NP \not\subseteq AC^0_4[2^{o(\sqrt{\log n})}]$.

3. If $AC^0 \cap \text{Mono} \subseteq \text{mSIZE}[\text{poly}]$, then $NC^2 \not\subseteq NC^1$.

4. If $NC^1 \cap \text{Mono} \subseteq \text{mSIZE}[2^{O(n^\varepsilon)}]$, then $NC^2 \not\subseteq NC^1$.

5. If $AC^0 \cap \text{Mono} \cap \text{GraphProperties} \subseteq \text{mSIZE}[\text{poly}]$, then $NP \not\subseteq NC^1$.

6. If $NC^1 \cap \text{Mono} \cap \text{GraphProperties} \subseteq \text{mSIZE}[\text{poly}]$, then $L \not\subseteq NC^1$.

Item (3) of Theorem 1.3 implies in particular that, if the upper bound of Theorem 1.2 cannot be improved to $AC^0$ (i.e., the question asked by [GS92] has a negative answer), then $NC^2 \not\subseteq NC^1$. It also improves a result from [CHO+20] showing the weaker conclusion $NP \not\subseteq NC^1$ under the same assumption.

Even if it’s impossible to efficiently simulate $AC^0$ circuits computing monotone functions using unbounded depth monotone circuits, it could still be the case that a simulation exists for certain classes of monotone functions with additional structure. As explained above, Rossman’s result [Ros08a, Ros17b] achieves this for graph properties that are preserved under homomorphisms. Items (5) and (6) of Theorem 1.3 show that a simulation that holds for all monotone graph properties is sufficient to get new separations in computational complexity.

### 1.1.3. Monotone complexity of constraint satisfaction problems

Recall that [GKRS19] showed the existence of a monotone function $f^{GKRS}$ in $NC^2$ that is not in $\text{mSIZE}[2^{O(1)}]$. As opposed to classical results [Raz85b, And85, AB87, Tar88] that rely on the approximation method, their monotone circuit lower bound employs a lifting technique from communication complexity. It is thus natural to consider if their approach can be adapted to provide a monotone function $g$ that is efficiently computable by constant-depth circuits but is not in $\text{mSIZE}[\text{poly}]$.

As remarked in [GKRS19, dRGR22], all monotone lower bounds obtained from lifting theorems so far also hold for monotone encodings of constraint satisfaction problems (CSPs). Next, we introduce a class of monotone Boolean functions $\text{CSP-SAT}_S$ which capture the framework and lower bound of [GKRS19].
**Encoding CSPs as monotone Boolean functions.** Let $R \subseteq \{0,1\}^k$ be a relation. We call $k$ the *arity* of $R$. Let $V = (i_1, \ldots, i_k) \in [n]^k$, and let $f_{R,V} : \{0,1\}^n \rightarrow \{0,1\}$ be the function that accepts a string $x \in \{0,1\}^n$ if $(x_{i_1}, \ldots, x_{i_k}) \in R$. We call $f_{R,V}$ a *constraint application* of $R$ on $n$ variables. (A different choice of the sequence $V$ gives a different constraint application of $R$.) If $S$ is a finite set of Boolean relations, we call any set of constraint applications of relations from $S$ on a fixed set of variables an *$S$-formula*. In particular, we can describe an $S$-formula through a set of pairs $(V,R)$. We say that an $S$-formula $F$ is *satisfiable* if there exists an assignment to the variables of $F$ which satisfies all the constraints of $F$.

Let $S = \{R_1, \ldots, R_k\}$ be a finite set of Boolean relations. Let $\ell_i$ be the arity of the relation $R_i$. Note that there are $n^{\ell_i}$ possible constraint applications of the relation $R_i$ on $n$ variables. Let $N := \sum_{i=1}^k n^{\ell_i}$. We can identify each $S$-formula $F$ on a fixed set of $n$ variables with a corresponding string $w_F \in \{0,1\}^N$, where $w_F^j = 1$ if and only if the $j$-th possible constraint application (corresponding to one of the $N$ pairs $(V,R)$) appears in $F$. Let $\text{CSP-SAT}_S : \{0,1\}^n \rightarrow \{0,1\}$ be the Boolean function which accepts a given $S$-formula $F$ if $F$ is unsatisfiable. Note that this is a monotone function. When $n$ is clear from the context or we view $\{\text{CSP-SAT}_S\}_{n \geq 1}$ as a sequence of functions, we simply write $\text{CSP-SAT}_S$.

The function $f_{\text{GRS}}$ from [GKRS19] is simply $\text{CSP-SAT}_S$ for $S = \{\oplus_3, \oplus_3\}$, where $\oplus_3(x_1, x_2, x_3) = 1$ if and only if $\sum_i x_i \equiv b \pmod{2}$. More generally, for any finite set $S$ of Boolean relations, their framework shows how to lift a Resolution width (resp. depth) lower bound for an arbitrary unsatisfiable $S$-formula $F$ over $m$ variables into a corresponding monotone circuit size (resp. depth) lower bound for $\text{CSP-SAT}_S$, where $n = \text{poly}(m)$.

Despite the generality of the technique from [GKRS19] and the vast number of possibilities for $S$, we prove that a direct application of their approach cannot establish Theorem 1.1 and Theorem 1.2. This is formalised as follows. (We refer to Section 5 for much stronger forms of the result.)

**Theorem 1.4** (Limits of the direct approach via lifting and CSPs). Let $S$ be a finite set of Boolean relations. The following holds.

1. If $\text{CSP-SAT}_S \notin \text{mSIZE}[\text{poly}]$ then $\text{CSP-SAT}_S$ is $\oplus$-hard under $\leq_{\text{AC}^0}^m$ reductions.

2. If $\text{CSP-SAT}_S \notin \text{mNC}^1$ then $\text{CSP-SAT}_S$ is $\text{L}$-hard under $\leq_{\text{AC}^0}^m$ reductions.

In particular, since there are functions (e.g., Majority) computable in logarithmic space that are not in $\text{AC}^0[\oplus]$, Theorem 1.4 (Part 2) implies that any CSP-SAT function that is hard for poly-size monotone formulas (mNC$^1$) must lie outside $\text{AC}^0[\oplus]$. Observe that this can also be interpreted as a *monotone simulation*: for any finite set $S$ of Boolean relations, if $\text{CSP-SAT}_S \in \text{AC}^0[\oplus]$ then $\text{CSP-SAT}_S \in \text{mNC}^1$.

Theorem 1.4 is a corollary of a general result that completely classifies the monotone circuit complexity of Boolean-valued constraint satisfaction problems based on the set $\text{Pol}(S)$ of *poly-morphisms* of $S$, a standard concept in the investigation of CSPs.\(^8\) We present next a simplified

---

\(^8\)Roughly speaking, $\text{Pol}(S)$ captures the amount of symmetry in $S$, and a larger set $\text{Pol}(S)$ implies that solving $\text{CSP-SAT}_S$ is computationally easier. We refer the reader to Section 5 for more details and for a discussion of Post’s lattice, which is relevant in the next statement.
version of this result, which shows a dichotomy for the monotone circuit size and depth of Boolean-valued constraint satisfaction problems. We refer to Section 5 for a more general formulation and additional consequences.

**Theorem 1.5** (Dichotomies for the monotone complexity of Boolean-valued CSPs). Let $S$ be a finite set of Boolean relations. The following holds.

1. **Monotone Size Dichotomy**: If $\text{Pol}(S) \subseteq L_3$ there is $\varepsilon > 0$ such that $\text{mSIZE}(\text{CSP-SAT}_S) = 2^{\Omega(n^{\varepsilon})}$. Otherwise, $\text{mSIZE}(\text{CSP-SAT}_S) = n^{O(1)}$.

2. **Monotone Depth Dichotomy**: If $\text{Pol}(S) \subseteq L_3$ or $\text{Pol}(S) \subseteq V_2$ or $\text{Pol}(S) \subseteq E_2$, there is $\varepsilon > 0$ such that $\text{mDEPTH}(\text{CSP-SAT}_S) = \Omega(n^{\varepsilon})$. Otherwise, $\text{CSP-SAT}_S \in \text{mNC}^2$.

We note that previous papers of Schaefer [Sch78] and Allender, Bauland, Immerman, Schnoor and Vollmer [ABI+09] provided a conditional classification of the complexity of such CSPs. Theorem 1.5 and its extensions, which build on their results and techniques, paint a complete and unconditional picture of their monotone complexity.9

### 1.2 Techniques

Our arguments combine in novel ways several previously unrelated ideas from the literature. The exposition below follows the order in which the results appear above, except for the overview of the proof of Theorem 1.1, which appears last. We discuss this result after explaining the proof of Theorem 1.2 and the classification of the monotone complexity of CSPs (Theorem 1.4 and Theorem 1.5), as this sheds light into how the proof of Theorem 1.1 was discovered and into the nature of the argument.

**A monotone circuit size lower bound for a function in $\text{AC}^0[\oplus]$.** We first give an overview of the proof of Theorem 1.2.

The lower bound of [GKRS19]. We begin by providing more details about the aforementioned monotone circuit lower bound of [GKRS19], since their result is a key ingredient in our separation (see [dRGR22] for a more detailed overview). Recall that their function $f^{\text{GKRS}}$ corresponds to $\text{CSP-SAT}_S$ for $S = \{\oplus_3, \oplus_3\}$. Following their notation, this is simply the Boolean function $3\text{-XOR-SAT}_n$: $\{0, 1\}^{2n^3} \rightarrow \{0, 1\}$ which uses each input bit to indicate the presence of a linear equation with exactly 3 variables. This (monotone) function accepts a given linear system over $\mathbb{F}_2$ if the system is unsatisfiable. As one of their main results, [GKRS19] employed a lifting technique from communication complexity to show the existence of a constant $\varepsilon > 0$ such that $\text{mSIZE}(3\text{-XOR-SAT}_n) = 2^{\varepsilon n}$. (We show in Appendix A that a weaker super-polynomial monotone circuit size lower bound for $3\text{-XOR-SAT}_n$ can also be obtained using the approximation method and a reduction.)

Sketch of the proof of Theorem 1.2. Since $3\text{-XOR-SAT}_n \in \text{NC}^2$ (see, e.g. [GKRS19]), their result implies that $\text{NC}^2 \cap \text{Mono} \nsubseteq \text{mSIZE}[2^{n^{\Omega(1)}}]$. On the other hand, we are after a separation between

---

9We remark that only recently has Schaefer’s classification been extended to the non-Boolean case [Zhu17, Bul17]. Though the refined classification of [ABI+09] is conjectured to hold analogously in the case of non-Boolean CSPs [LT09], this is still open (see the discussion in [Bul18, Section 7]).
constant-depth (non-monotone) circuits and polynomial-size (unbounded depth) monotone circuits. There are two natural ways that one might try to approach this challenge, as discussed next.

First, the lifting framework explored by [GKRS19] offers in principle the possibility that by carefully picking a different set $S$ of Boolean relations, one might be able to reduce the non-monotone depth complexity of CSP-SAT$_S$ while retaining super-polynomial monotone hardness. However, Theorem 1.4 shows that this is impossible, as explained above.

A second possibility is to combine the exponential $2^{n^2}$ monotone circuit size lower bound for 3-XOR-SAT$_n$ and a padding argument, since we only need super-polynomial hardness. Indeed, this argument can be used to define a monotone function $g: \{0,1\}^n \rightarrow \{0,1\}$ that is computed by polynomial-size fan-in two circuits of depth poly(log log $n$) but requires monotone circuit of size $n^{\omega(1)}$. However, it is clear that no padding argument alone can reduce the non-monotone circuit depth bound to $O(1)$ while retaining the desired monotone hardness.

Given that both the classical widely investigated approximation method for monotone lower bounds and the more recent lifting technique do not appear to work in their current forms, for some time it seemed to us that, if true, a significantly new technique would be needed to establish a separation similar to the one in Theorem 1.2.

Perhaps surprisingly, it turns out that a more clever approach that combines padding with a non-trivial circuit upper bound can be used to obtain the result. The first key observation, already present in [GKRS19] and other papers, is that 3-XOR-SAT$_n$ can be computed not only in NC$^2$ but actually by polynomial-size span programs over $\mathbb{F}_2$. On the other hand, it is known that this model is equivalent in power to parity branching programs [KW93], which correspond to the non-uniform version of $\oplus$L, i.e., counting modulo 2 the number of accepting paths of a nondeterministic Turing machine that uses $O(\log n)$ space. A second key idea is that such a computation can be simulated by AC$^0[\oplus]$ circuits of sub-exponential size and large depth. More precisely, similarly to an existing simulation of NL (nondeterministic logspace) by AC$^0$ circuits of depth $d$ and size $2^{n^{O(1/d)}}$ via a “guess-and-verify” approach, it is possible to achieve an analogous simulation of $\oplus$L using AC$^0[\oplus]$ circuits (this folklore result appears implicit in [AKR+01] and [OSS19]). Putting everything together, it follows that for a large enough but constant depth, 3-XOR-SAT$_n$ can be computed by AC$^0[\oplus]$ circuits of size $2^{n^{\epsilon/2}}$. Since this function is hard against monotone circuits of size $2^{n^\epsilon}$, a padding argument can now be used to establish a separation between AC$^0[\oplus]$ and mSIZE[poly]. (A careful choice of parameters provides the slightly stronger statement in Theorem 1.2.)

**Non-trivial monotone simulations and their consequences.** In order to conclude that significantly stronger monotone simulations imply new complexity separations (Theorem 1.3), we argue contrapositively. By supposing a complexity collapse, we can exploit known monotone circuit lower bounds to conclude that a hard monotone function exists in a lower complexity class. For instance, if NC$^2 \subseteq$ NC$^1$, then 3-XOR-SAT $\in$ NC$^1$, and we can conclude by standard depth-reduction for NC$^1$ and padding, together with the exponential lower bound for 3-XOR-SAT due to [GKRS19], that there exists a monotone function in AC$^0$ which is hard for polynomial-size monotone circuits. The other implications are argued in a similar fashion. In particular, we avoid the more complicated use of hardness magnification from [CHO+20] to establish this kind of result, while also getting a stronger consequence.

A little more work is required in the case of graph properties (Theorem 1.3 Items 5 and 6), as padding the function computing a graph property does not yield a graph property. We give a general lemma that allows us to pad monotone graph properties while preserving their struc-
ture (Lemma 3.6). We then argue as in the case for general functions, using known monotone lower bounds for graph properties. We note that Lemma 3.6 is also important in the proof of Theorem 1.1, which will be discussed below. We believe that our padding technique for graph properties might find additional applications.

**Monotone complexity of CSPs.** These are the most technical results of the paper. Since explaining the corresponding proofs requires more background and case analysis, here we only briefly describe the main ideas and references behind Theorem 1.4, Theorem 1.5, and the extensions discussed in Section 5.

A seminal work of Schaefer [Sch78] proved that any Boolean CSP is either solvable in polynomial-time or it is NP-complete. Later, Jeavons [Jea98] observed that the complexity of deciding if a given set of constraint applications of $S$ is satisfiable depends exclusively on the set $\text{Pol}(S)$ of polymorphisms of $S$. Intuitively, the set of polymorphisms of a set of relations is a measure of its symmetry. The more symmetric a set of relations is, the lesser is its expressive power. Jeavons formally proves this intuition by showing that, if $\text{Pol}(S) \subseteq \text{Pol}(S')$, then the problem of deciding the satisfiability of a given $S'$-formula can be reduced in polynomial-time to that of deciding the satisfiability of a given $S$-formula. This allows Jeavons to reprove Schaefer’s result.

Existing proofs and classification results for constraint satisfaction problems do not encode the satisfiability problem as a monotone Boolean function $\text{CSP-SAT}_S$, in the way we described above. We reexamine Schaefer’s and Jeavons’s proofs and establish that the reduction from $\text{CSP-SAT}_{S'}$ to $\text{CSP-SAT}_S$ can also be done with efficient monotone circuits. Making use of and adapting parts of the refined results and analysis of [ABI+09], which builds on the earlier dichotomy result of [Sch78] and provides a detailed picture of the computational complexity of Boolean-valued CSPs, we prove in fact that the underlying reductions can all be done in monotone nondeterministic logspace.

Finally, using known upper and lower bounds for monotone circuits together with a direct analysis of some basic cases, and inspecting Post’s lattice [Pos41, BCRV03, BCRV04], we are able to show that $\text{CSP-SAT}_S$ is hard for monotone circuits only when $\text{CSP-SAT}_S$ is $\oplus L$-complete, as in Theorem 1.4 Part 1.

**A monotone circuit depth lower bound for a function in $\text{AC}^0$.** Next, we combine insights obtained from the monotone lower bound of [GKRS19], our proof of Theorem 1.2 via a guess-and-verify depth reduction and padding, and the statement of Theorem 1.4 (limits of the direct approach via CSPs) to get the separation in Theorem 1.1. As alluded to above, our approach differs from those of [Oko82, AG87, BST13, COS17] and related results in the context of first-order logic [Sto95, Kup21, Kup22].

Recall that the [GKRS19] framework lifts a Resolution width lower bound for an unsatisfiable $S$-formula $F$ into a corresponding monotone circuit size lower bound for $\text{CSP-SAT}_S$. On the other hand, Theorem 1.4 rules out separating constant-depth circuits from monotone circuits of polynomial size via $\text{CSP-SAT}_S$ functions. In particular, we cannot directly apply the chain of reductions from [GKRS19] to obtain the desired separation result. Instead, we extract from the specific $S$-formula $F$ that they use a structural property that will allow us to improve the $\text{AC}^0[\oplus]$ upper bound from Theorem 1.2 to the desired $\text{AC}^0$ upper bound in Theorem 1.1.

In [GKRS19] the formula $F$ is a Tseitin contradiction, a well-known class of unsatisfiable CNFs with a number of applications in proof complexity. For an undirected graph $G$, the Tseitin formula $T(G)$ encodes a system of linear equations modulo 2 as follows: each edge $e \in E(G)$ becomes a
Boolean variable \( x_e \), and each vertex \( v \in V(G) \) corresponds to a constraint (linear equation) \( C_v \)

stating that \( \sum_{u \in N_G(v)} x_{\{v,u\}} = 1 \pmod{2} \), where \( N_G(v) \) denotes the set of neighbours of \( v \) in \( G \).

Crucially, \( T(G) \) does not encode an arbitrary system of linear equations, i.e., the following key structural property holds: every variable \( x_e \) appears in exactly 2 equations.

On a technical level, this property is not preserved when obtaining a (total) monotone function CSP-SAT\(_S\) by the gadget composition employed in the lifting framework and its reductions. However, we can still hope to explore this property in a somewhat different argument with the goal of obtaining CSP instances that lie in a complexity class weaker than \( \oplus L \), which is the main bottleneck in the proof of Theorem 1.2 yielding \( \text{AC}^0[\oplus] \) circuits instead of \( \text{AC}^0 \). At the same time, considering this structural property immediately takes us outside the domain of Theorem 1.4, which does not impose structural conditions over the CSP instances.

We can capture the computational problem corresponding to this type of system of linear equations using the following Boolean function. Let \( \text{OddFactor}_n : \{0,1\}^{\binom{n}{2}} \to \{0,1\} \) be the function that accepts a given graph \( G \) if the formula \( T(G) \) described above is satisfiable. (Equivalently, if \( G \) admits a spanning subgraph in which the degree of every vertex is odd.) Note that \( \text{OddFactor}_n \) is a monotone Boolean function: adding edges to \( G \) cannot make a satisfiable system unsatisfiable, since we can always set a new edge variable \( x_e \) to 0.

While 3-XOR-SAT (the corresponding CSP-SAT\(_S\) function obtained from an appropriate Tseitin formula via the framework of [GKRS19]) admits a \( \oplus L \) upper bound, we observe that \( \text{OddFactor}_n \) can be computed in \( L \) thanks to its more structured class of input instances. Indeed, one can prove that the formula \( T(G) \) is satisfiable if and only if every connected component of \( G \) has an even number of vertices.\(^{10}\) In turn, the latter condition can be checked in logarithmic space using Reingold’s algorithm for undirected s-t-connectivity [Rei05]. (We note that related ideas appear in an unpublished note of Johannsen [Joh03].) This is the first application of Reingold’s algorithm to this kind of separation.

At the same time, \( \text{OddFactor}_n \) retains at least part of the monotone hardness of 3-XOR-SAT. Using a different reduction from a communication complexity lower bound, [BGW99] proved that the monotone circuit depth of \( \text{OddFactor}_n \) is \( n^{\Omega(1)} \). Altogether, we obtain a monotone Boolean function (indeed a graph property) that lies in \( L \) but is not in \( \text{mDEPTH}[\text{log}^6 n] \). Applying a guess-and-verify depth reduction for \( L \) and using (graph) padding (analogously to the proof sketch of Theorem 1.2), we get a monotone graph property in \( \text{AC}^0 \) that is not in \( \text{mDEPTH}[\text{log}^6 n] \). This completes the sketch of the proof of Theorem 1.1.

1.3 Directions and open problems

Constant-depth circuits and monotone circuits are possibly the two most widely investigated models in circuit complexity theory. Although our results provide new insights about the relation between them, there are exceptionally basic questions that remain open.

While [Qui53] showed that negations can be efficiently eliminated from circuits of depth \( d \leq 2 \) that compute monotone functions, already at depth \( d = 3 \) the situation is much less clear. Theorem 4.3 (see Section 4.1) implies that every monotone function in depth-3 \( \text{AC}^0 \) admits a monotone circuit of size \( 2^{n-\Omega(n/\log^2 n)} \). It is unclear to us if this is optimal. While [COS17]

\(^{10}\)A simple parity argument shows that odd-sized components cannot be satisfied. On the other hand, we can always satisfy an even-sized component by starting with an arbitrary assignment, which must satisfy an even number of constraints by a parity argument, and flipping the values of the edges in a path between unsatisfied nodes, until all nodes in the connected component are satisfied.
rules out an efficient constant-depth monotone simulation, it is still possible (and consistent with Theorem 1.1) that $\text{AC}^0_3 \cap \text{Mono} \subseteq \text{mNC}^1$. Is there a significantly better monotone circuit size upper bound for monotone functions computed by polynomial-size depth-3 circuits?

Our results come close to solving the question posed by Grigni and Sipser [GS92]. Using our approach, it would be sufficient to show that OddFactor, requires monotone circuits of size $\exp(n^{\Omega(1)})$. This is closely related to the challenge of obtaining an exponential monotone circuit size lower bound for Matching, a longstanding open problem in monotone complexity (see [Juk12, Section 9.11]). Indeed, it’s possible to reduce OddFactor to Matching using monotone AC$^0$ circuits (see [AK11, Lemma 6.18]).

Incidentally, the algebraic complexity variant of the AC$^0$ vs. mSIZE[poly] problem has been recently settled in a strong way through a new separation result obtained by Chattopadhyay, Datta, and Mukhopadhyay [CDM21]. Could some of their techniques be useful to attack the more elusive Boolean case?

Finally, it would be interesting to develop a more general theory able to explain when cancellations can speedup the computation of monotone Boolean functions. Our investigation of monotone simulations and separations for different classes of monotone functions (graph properties and constraint satisfaction problems) can be seen as a further step in this direction.

Acknowledgements. We thank Arkadev Chattopadhyay for several conversations about the AC$^0$ versus mSIZE[poly] problem and related questions. We are also grateful to Denis Kuperberg for explaining to us the results from [Kup21, Kup22]. The first author thanks Ninad Rajgopal for helpful discussions about depth reduction. Finally, we thank Gernot Salzer for the code used to generate Figures 1, 2, and 3. This work received support from the Royal Society University Research Fellowship URF\R1\191059, the EPSRC New Horizons Grant EP/V04201/1, and the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick.

2 Preliminaries

2.1 Notation

Boolean functions. We denote by Mono the set of all monotone Boolean functions. We define $\text{poly} = \{ n \mapsto n^C : C \in \mathbb{N} \}$. A Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is said to be a graph property if $f(G) = f(H)$ for any two isomorphic graphs $G$ and $H$. Let $\mathcal{F} = \{ f_n \}_{n \in \mathbb{N}}$ be a sequence of graph properties, where $f_n$ is defined over undirected graphs on $n$ vertices. We say that $\mathcal{F}$ is preserved under homomorphisms if, whenever there is a homomorphism from a graph $G$ to a graph $H$, we have $\mathcal{F}(G) \leq \mathcal{F}(H)$. We denote by HomPreserving the set of all graph properties which are preserved under homomorphisms. Note that HomPreserving $\subseteq$ Mono.

Boolean circuits. We denote by AC$^0_d[s]$ the family of Boolean functions computed by size-$s$, depth-$d$ Boolean circuits with unbounded fan-in $\{ \land, \lor \}$-gates and input literals from $\{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n} \}$. We write AC$^0[\text{poly}]$ as a shorthand for $\bigcup_{d=0}^{\infty} \text{AC}^0_d[\text{poly}]$, and AC$^0$ as a shorthand of AC$^0[n^{O(1)}] = \text{AC}^0[\text{poly}]$. We will also refer to AC$^0_d[\text{poly}]$ by AC$^0_d$. We write DNF$[s]$ to denote the family of Boolean functions computed by size-$s$ DNFs, where size is measured by number of terms. We write CNF$[s]$ analogously.

\[\text{Note that in OddFactor we are concerned with the existence of a spanning subgraph where the degree of every vertex is odd, while in Matching the degree should be exactly 1.}\]
We write \( \text{SIZE}[s] \) to denote the family of Boolean functions computed by size-\( s \) circuits. We write \( \text{DEPTH}[d] \) to denote the family of Boolean functions computed by fan-in 2 circuits of depth \( d \). We denote by \( \text{AC}^0[\oplus] \) the family of Boolean functions computed by polynomial-size \( \text{AC}^0 \) circuits with unbounded fan-in \( \oplus \)-gates.

We denote by \( L \) the family of Boolean functions computed by logspace machines, and by \( \text{NL} \) the family of Boolean functions computed by polynomial-time nondeterministic logspace machines. Moreover, we denote by \( \oplus L \) the family of Boolean functions computed by polynomial-time nondeterministic logspace machines with a parity acceptance condition (i.e., an input is accepted if the number of accepting paths is odd).

Circuit complexity. Given a circuit class \( \mathcal{C} \), we write \( m\mathcal{C} \) to denote the monotone version of \( \mathcal{C} \). Given a function \( f \), we write \( m\text{SIZE}(f) \) to denote the size of the smallest monotone circuit computing \( f \) and \( m\text{DEPTH}(f) \) to denote the smallest depth of a fan-in 2 monotone circuit computing \( f \). Given two Boolean functions \( f, g \), we write \( f \leq_m \text{Proj} g \) if there exists a many-one reduction from \( f \) to \( g \) in which each bit of the reduction is a monotone projection\(^{12} \) of the input.

Miscellanea. Let \( \alpha \in \{0,1\}^n \). We define \( |\alpha|_1 := \sum_{i=1}^n \alpha_i \). We call \( |\alpha|_1 \) the Hamming weight of \( \alpha \). We let \( \text{supp}(\alpha) = \{i \in [n] : \alpha_i = 1\} \). We let \( \text{THR}_{k,n} : \{0,1\}^n \rightarrow \{0,1\} \) be the Boolean function such that \( \text{THR}_{k,n}(x) = 1 \iff |x|_1 \geq k \).

### 2.2 Background results

The next lemma, which is proved via a standard “guess-and-verify” approach, shows that nondeterministic logspace computations can be simulated by circuits of size \( 2^{n^\varepsilon} \) and of depth \( d = O(1) \).

**Lemma 2.1** (Folklore; see, e.g., [AHM+08, Lemma 8.1]). For all \( \varepsilon > 0 \), we have \( \text{NL} \subseteq \text{AC}^0[2^{n^\varepsilon}] \).

### 3 Constant-Depth Circuits vs. Monotone Circuits

In this section, we prove Theorems 1.1 and 1.2. For the upper bounds, we require the logspace graph connectivity algorithm due to [Rei05] and the \( \oplus L \) algorithm for solving linear systems over \( \mathbb{F}_2 \) due to [BDHM92], as well as the depth-reduction techniques of [AKR+01, AHM+08]. On the lower bounds side, our proofs rely on previous monotone circuit and depth lower bounds from [BGW99, GKRS19]. In order to obtain a monotone formula lower bound for a graph property, we prove a graph padding lemma in Section 3.2.

#### 3.1 A monotone size lower bound for a function in \( \text{AC}^0[\oplus] \)

In this section, we prove Theorem 1.2. We first recall the monotone circuit lower bound of [GKRS19] and a depth-reduction lemma implicit in [AKR+01] and [OSS19], whose full proof we give below for completeness. We remark that similar arguments can be employed to prove Lemma 2.1, essentially by replacing the \( \oplus \) gates by \( \lor \) gates.

As explained in Section 1.2, in its strongest form the separation result from [GKRS19] can be stated as follows.

\(^{12}\)A monotone projection is a projection without negations.
We may also assume that each configuration in the configuration graph $G$ of $M$ is time-stamped – in other words, each configuration carries the information of the number of computational steps it takes to arrive at it.\footnote{Formally, we can define a $\oplus$-machine $M'$ such that the configurations of $M'$ are $(C, t)$, where $C$ is a configuration of $M$, and $t = 0, 1, \ldots, m = n^{O(1)}$ is a number denoting the time in which the configuration was achieved. A configuration $(C, t)$ can only reach a configuration $(C', t + 1)$ in the configuration graph of $M'$.} We may also assume that every accepting computation takes exactly the same amount of time, which means that every path from the starting configuration $v_{\text{start}}$ to the accepting configuration $v_{\text{accept}}$ has the same length in the configuration graph. These assumptions imply that the configuration graph is layered (because a configuration with time-stamp $t$ can only point to configurations with time-stamp $t + 1$) and acyclic. Note that, for a fixed machine, the configuration graph can be computed from the input string using a projection.

Let $m = n^{O(1)}$ be the time that an accepting computation takes. We now show how to count (modulo 2) the number of accepting paths from $v_{\text{start}}$ to $v_{\text{accept}}$ with a depth-$d$ $\text{AC}^0[\oplus]$ circuit. First, choose $m^{1/d} - 1$ configurations $v_1, \ldots, v_{m^{1/d} - 1}$ (henceforth called “checkpoints”) from $V(G)$, such that the configuration $v_i$ is at the level $i \cdot m^{1-1/d}$ in the configuration graph (i.e., it takes $i \cdot m^{1-1/d}$ time steps to arrive at $v_i$). For convenience, we let $v_0 = v_{\text{start}}$ and $v_{m^{1/d}} = v_{\text{accept}}$. We then count the number of paths from from $v_{\text{start}}$ to $v_{\text{accept}}$ that go through $v_1, \ldots, v_{m^{1/d} - 1}$, and sum over all possible choices of the checkpoints. Since the graph is layered and each path from $v_0$ to $v_{m^{1/d}}$ has length exactly $m$, there is only one choice of checkpoints that witnesses a given path from $v_0$ to $v_{m^{1/d}}$, so no path is counted twice in this summation. Letting $\#\text{paths}(s, t, \ell)$ denote the number of paths between configurations $s$ and $t$ with distance exactly $\ell$, we obtain

$$\#\text{paths}(v_0, v_{m^{1/d}}, m) = \sum_{v_1, \ldots, v_{m^{1/d} - 1}} \prod_{i=0}^{m^{1/d} - 1} \#\text{paths}(v_i, v_{i+1}, m^{1-1/d}).$$

The above calculation can be done in modulo 2 with an unbounded fan-in XOR gate (replacing the summation) and an unbounded fan-in AND gate (replacing the product). Note that the formula above is recursive. Repeating the same computation for calculating (modulo 2) the expression $\#\text{paths}(v_i, v_{i+1}, m^{1-1/d})$ for each $i$, we obtain a depth-$2d$ $\text{AC}^0[\oplus]$ circuit for calculating the number of paths from $v_{\text{start}}$ to $v_{\text{accept}}$ (modulo 2). Clearly, the total size of the circuit is $2^{O(m^{1/d} \log m)}$, which is smaller than $2^{n^\delta}$ for a large enough constant $d$.

We now restate Theorem 1.2 and prove it by combining Theorem 3.1 and Lemma 3.2 with a padding trick.

**Theorem 1.2** (Polynomial-size constant-depth vs. larger monotone size). For every $k \geq 1$, we have $\text{AC}^0[\oplus] \cap \text{Mono} \not\subseteq \text{mSIZE}[2^{\log n^k}]$.\[13\]
Proof. By Theorem 3.1, there exists $\varepsilon > 0$ and a monotone function $f \in \oplus L$ such that any monotone circuit computing $f$ has size $2^{\Omega(n^\varepsilon)}$.

Let $\delta = \varepsilon/k$ and let $m = 2^{n\delta}$. Let $g : \{0, 1\}^n \times \{0, 1\}^m \to \{0, 1\}$ be the Boolean function defined as $g(x, y) = f(x)$. Note that $g$ is a function on $N := m + n = 2^{O(n^\delta)}$ bits. By Lemma 3.2, there exists an $AC^0[\oplus]$ circuit computing $f$ of size $2^n\delta = N^{O(1)}$. The same circuit computes $g$. On the other hand, any monotone circuit computing $g$ has size $2^{\Omega(n^\delta)} = 2^{\Omega((\log N)^{\varepsilon/\delta})} = 2^{\Omega((\log N)^k)}$. \dashbox{0}

3.2 A monotone depth lower bound for a graph property in $AC^0$

In this section, we prove Theorem 1.1. We prove moreover that the function that separates $AC^0 \cap \text{Mono}$ and $\text{mNC}^i$ can be taken to be a graph property. We state our result in its full generality below.

**Theorem 3.3.** For every $i \geq 1$, we have $AC^0 \cap \text{Mono} \cap \text{GraphProperties} \not\subseteq \text{mDEPTH}[\log(n)^i]$. In particular, we have $AC^0 \cap \text{Mono} \cap \text{GraphProperties} \not\subseteq \text{mNC}^i$.

First, we recall a result of [BGW99], which proves monotone lower bounds for the following function. Let $\text{OddFactor}_n : \{0, 1\}^{\binom{n}{2}} \to \{0, 1\}$ be the function that accepts a given graph if it contains an odd factor – in other words, a spanning subgraph in which the degree of every vertex is odd. Babai, Gál and Wigderson [BGW99] proved the following result:

**Theorem 3.4 ([BGW99]).** Any monotone formula computing $\text{OddFactor}_n$ has size $2^{\Omega(n)}$, and any monotone circuit computing $\text{OddFactor}_n$ has size $n^{\Omega(\log n)}$.

The proof in [BGW99] is actually for the case of bipartite graphs, but it easily extends to general graphs, since the bipartite case reduces to the general case by a monotone projection. The formula lower bound stated above is slightly stronger because it makes use of asymptotically optimal lower bounds on the randomized communication complexity of $\text{DISJ}_n$ [KS92], which were not available to [BGW99]. We remark that, with a different language, a monotone circuit lower bound for $\text{OddFactor}$ is also implicitly proved in Feder and Vardi [FV98, Theorem 30].

We now recall an upper bound for $\text{OddFactor}$, implicitly proved in an unpublished note due to Johannsen [Joh03].

**Theorem 3.5 ([Joh03]).** We have $\text{OddFactor} \in L$.

**Proof.** We first recall the following observation about the $\text{OddFactor}$ function, which appears in different forms in the literature (see [Urq87, Lemma 4.1] or [Juk12, Lemma 18.16]; see also [Joh03, Proposition 1] for a different proof.)

**Claim.** A graph $G$ has an odd factor if and only if every connected component of $G$ has an even number of vertices.

**Proof.** If a graph $G$ has an odd factor, we can conclude that every connected component of $G$ has an even number of vertices from the well-known observation that in every graph there is an even number of vertices of odd degree.

Now suppose that every connected component of $G$ has an even number of vertices. We will iteratively construct an odd factor $F$ of $G$. We begin with the empty graph. We take any two vertices $u, v$ in the same connected component of $G$ which currently have even degree in $F$, and consider any path $P = (x_1, \ldots, x_k)$ between $u$ and $v$, where $x_1 = u$ and $x_k = v$. If the edge $x_i \overline{x}_{i+1}$
is currently in \( F \), we remove \( x_i x_{i+1} \) from \( F \); otherwise, we add \( x_i x_{i+1} \) to \( F \). It’s easy to check that, in every iteration of this procedure, only the vertices \( u \) and \( v \) have the parity of their degree changed in \( F \); the degree of every other vertex stays the same (modulo 2). Since every connected component has an even number of vertices, this means that, eventually, every vertex in \( F \) will have odd degree.

Now it’s easy to check in logspace if every connected component of \( G \) has an even number of vertices using Reingold’s algorithm for undirected connectivity [Rei05]. It suffices to check if, for every vertex \( v \) of \( G \), the number of vertices reachable from \( v \) is odd.

Now, if we only desire to obtain a function in \( \text{AC}^0 \) not computed by monotone circuits of depth \( (\log n)^i \), we can follow the same argument of Theorem 1.2, using Lemma 2.1 instead of Lemma 3.2. In order to obtain moreover a monotone graph property witnessing this separation, we will need the following lemma, which enables us to obtain a graph property after “padding” a graph property. We defer the proof of this lemma to the end of this section.

Lemma 3.6. Let \( f : \{0, 1\}^{(2)} \to \{0, 1\} \) be a monotone graph property on graphs of \( n \) vertices. The following holds.

1. If \( f \in \text{NC}^i \) for some \( i > 1 \), then there exists a monotone graph property \( g \) on graphs of \( N = 2^{(\log n)^i} \) vertices such that \( g \in \text{NC}^1 \) and \( f \leq \text{mProj}^m g \).

2. If \( f \in \text{NL} \), then for all \( \varepsilon > 0 \) there exists a monotone graph property \( g \) on graphs of \( N = 2^{n^\varepsilon} \) vertices such that \( g \) can be computed by \( \text{AC}^0 \) circuits of size \( N^{2+o(1)} \) and \( f \leq \text{mProj}^m g \).

3. If \( f \in \oplus \text{L} \), then for all \( \varepsilon > 0 \) there exists a monotone graph property \( g \) on graphs of \( N = 2^{n^\varepsilon} \) vertices such that \( g \) can be computed by \( \text{AC}^0[\oplus] \) circuits of size \( N^{2+o(1)} \) and \( f \leq \text{mProj}^m g \).

We are now ready to prove Theorem 3.3.

Proof of Theorem 3.3. Fix \( n \in \mathbb{N} \) and take an \( \varepsilon < 1/i \). Observing that \( \text{L} \subseteq \text{NL} \), from Theorem 3.5 and item (2) of Lemma 3.6 we conclude that there exists a monotone graph property \( f \) on \( N = 2^{n^\varepsilon} \) vertices such that \( f \in \text{AC}^0 \) and \( \text{OddFactor}_n \leq \text{mProj}^m f \). By Theorem 3.4, any monotone circuit computing \( f \) has depth \( \Omega(n) = \Omega((\log N)^{1/\varepsilon}) \gg (\log N)^i \).

Raz and Wigderson [RW92] observed that there exists a monotone function \( f \in \text{NC}^1 \setminus \text{mNC} \). Using Lemma 3.6, we observe moreover that it’s possible to obtain this separation with a monotone graph property.

Proposition 3.7. We have \( \text{NC}^1 \cap \text{Mono} \cap \text{GraphProperties} \not\subseteq \text{mNC} \).

Proof. Observing that \( \text{L} \subseteq \text{NC}^2 \), we conclude from Theorem 3.5 and item (1) of Lemma 3.6 that there exists a monotone graph property \( f \) on \( N = 2^{(\log n)^2} \) vertices such that \( f \in \text{NC}^1 \) and \( \text{OddFactor}_n \leq \text{mProj}^m f \). By Theorem 3.4, any monotone circuit computing \( f \) has depth \( \Omega(n) = \Omega(2^{\sqrt{\log N}}) \), which implies \( f \notin \text{mNC} \).
3.3 Efficient monotone padding for graph properties

We will now prove Lemma 3.6. We first recall some low-depth circuits for computing threshold functions, which we will use to design a circuit for efficiently computing the adjacency matrix of induced subgraphs.

**Theorem 3.8** ([HWWY94]). Let $d > 0$ be a constant. The function $\text{THR}_{(\log n)^d, n}$ can be computed by an $\text{AC}^0$ circuit of size $n^{o(1)}$ and depth $d + O(1)$.

**Theorem 3.9** ([AKS83]). For every $k \in [n]$, the function $\text{THR}_{k, n}$ can be computed by a circuit of depth $O(\log n)$ and size $n^{O(1)}$.

**Lemma 3.10.** There exists a circuit $C^k_n$ with $(\binom{n}{2}) + n$ inputs and $\binom{k}{2}$ outputs which, when given as input an adjacency matrix of a graph $G$ on $n$ vertices and a characteristic vector of a set $S \subseteq [n]$ such that $|S| \leq k$, outputs the adjacency matrix of the graph $G[S]$, padded with isolated vertices when $|S| < k$. The circuit has constant-depth and size $n^{2+o(1)}$ when $k = \text{polylog}(n)$, and size $n^{O(1)}$ and depth $O(\log n)$ otherwise.

**Proof.** Let $\{x_{ij}\}_{i,j \in [n]}$ encode the adjacency matrix of $G$. Let $\alpha \in \{0,1\}^n$ be the characteristic vector of $S$. Let $i, j \in [k]$. Note that $\{i, j\} \in E(G[S])$ if and only if there exists $a, b \in [n]$ such that

- $\alpha_a$ is the $i$-th non-zero entry of $\alpha$,
- $\alpha_b$ is the $j$-th non-zero entry of $\alpha$, and
- $x_{ab} = 1$ (i.e., $a$ and $b$ are connected in $G$).

We first consider the case $k = \text{polylog}(n)$. In this case, the first two conditions can be checked with circuits of size $n^{o(1)}$ using Theorem 3.8. Therefore, we can compute if $i$ and $j$ are adjacent using $n^{2+o(1)}$ gates and constant depth. As there are at most $(\log n)^{O(1)}$ such pairs, we can output $G[S]$ with at most $n^{2+o(1)}$ gates.

For any $k$, the first two conditions can be checked with an $\text{NC}^1$ circuit by Theorem 3.9. Since there are at most $n^2$ pairs $i, j$, the entire adjacency matrix can be computed with a $O(\log n)$-depth and polynomial-size circuit.

We are ready to prove Lemma 3.6.

**Proof of Lemma 3.6.** We first prove (1). Fix $n \in \mathbb{N}$ and let $N = 2^{(\log n)^d}$. For a graph $G$ on $N$ vertices such that $|E(G)| \leq \binom{n}{2}$, let $G_{\text{clean}}$ be the graph obtained from $G$ by removing isolated vertices from $G$ one-by-one, in lexicographic order, until one of the following two conditions are satisfied: (1) there are no more isolated vertices in $G_{\text{clean}}$, or (2) $G_{\text{clean}}$ has exactly $n$ vertices. Let $g : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ be the monotone graph property defined as follows:

$$g(G) := \left(|E(G)| > \binom{n}{2}\right) \lor (|V(G_{\text{clean}})| > n) \lor (f(G_{\text{clean}}) = 1).$$

Note that $g$ accepts a graph $G$ if and only if at least one of the following three conditions are satisfied:

1. $G$ has at most $\binom{n}{2}$ edges, $G_{\text{clean}}$ has exactly $n$ vertices and $f(G_{\text{clean}}) = 1$, or
2. $G$ has more than $\binom{n}{2}$ edges, or
3. $G_{\text{clean}}$ has more than $n$ vertices.

We observe that the monotonicity of $g$ follows from the monotonicity of $f$. We also claim that $g$ is a graph property. Indeed, the graph $G_{\text{clean}}$ is the same (up to isomorphism), irrespective of the order according to which the isolated vertices are removed from $G$. Moreover, the function $f$ is also a graph property. Because of this, all the three conditions above are preserved under isomorphisms.

We first observe that $f$ is a monotone projection of $g$. Indeed, given a graph $G$ on $n$ vertices, we can easily construct a monotone projection a graph $G'$ on $N$ vertices and at most $\binom{n}{2}$ edges such that $f(G) = g(G')$. We just let $G'$ have a planted copy of $G$, and all other vertices are isolated. Then $G'_{\text{clean}} = G$ (up to isomorphism) and $g(G') = f(G_{\text{clean}}) = f(G)$.

We now show how to compute $g$ in $\text{NC}^1$. Let $\{x_{ij}\}_{i,j \in [N]}$ be the input bits of $g$, corresponding to the adjacency matrix of a graph $G$. The circuit computes as follows.

1. If $|E(G)| > \binom{n}{2}$, accept the graph $G$.
2. Compute the characteristic vector $\alpha \in \{0,1\}^N$ of the set of all non-isolated vertices of $G$. If $|\alpha|_1 > n$, accept the graph $G$.
3. Compute $G_{\text{clean}}$ and output $f(G_{\text{clean}})$.

Note that checking if $|E(G)| > \binom{n}{2}$ can be done in $\text{NC}^1$ by Theorem 3.9. Moreover, for all $i \in [N]$, we have $\alpha_i = \bigvee_{j \in [N]} x_{ij}$, and therefore $\alpha_i$ can be computed by a circuit of depth $O(\log N)$ and $O(N)$ gates. In total, the vector $\alpha$ can be computed with $O(N^2)$ gates and $O(\log N)$ depth. Finally, we can check if $|\alpha|_1 > n$ in $\text{NC}^1$ with a threshold circuit.

For the final step, we compute $G_{\text{clean}}$. If $|\alpha|_1 = n$, note that $G_{\text{clean}} = G[\text{supp}(\alpha)]$. When $|\alpha|_1 < n$, then $G_{\text{clean}}$ is $G[\text{supp}(\alpha)]$ padded with isolated vertices. We can therefore compute $G_{\text{clean}}$ with the circuit $G_{\text{N}}^n$ of Lemma 3.10. Moreover, since $f \in \text{NC}^1$, we have that $f$ can be computed by a circuit of size $n^{O(1)} = N^{o(1)}$ and depth $O((\log n)\log N) = O(\log N)$. Therefore, computing $f(G_{\text{clean}})$ can be done in $\text{NC}^1$. Overall, we get that $g \in \text{NC}^1$.

In order to prove (2), it suffices to modify the proof above. The modification can be briefly described as follows. Let $N = 2^{\alpha \varepsilon}$. Every time Lemma 3.10 is applied, we use the $\text{AC}^0$ circuit instead of the $\text{NC}^1$ circuit, since $n = \text{polylog}(N)$. This amounts to $N^{2+o(1)}$ many gates with unbounded fan-in. Moreover, since by assumption $f \in \text{NL}$, applying Lemma 2.1 we obtain an $\text{AC}^0$ circuit for $f$ of size $2^{\alpha \varepsilon/2} = N^{o(1)}$, so we can compute $f(G_{\text{clean}})$ in constant depth with $N^{o(1)}$ gates.

Finally, for (3) it suffices to apply the same argument used for (2), replacing an application of Lemma 2.1 by an application of Lemma 3.2.

4 Non-Trivial Monotone Simulations and Their Consequences

In contrast to Section 3, in this section we observe that a non-trivial simulation of $\text{AC}^0$ circuits by monotone circuits is possible. This follows from a refined version of the switching lemma proved by Rossman [Ros17a]. As a proof of concept, we use this simulation result to reprove a well-known $\text{AC}^0$ lower bound for Majority.

In the second part of this section, we show that if much faster simulations are possible, then even stronger non-monotone circuit lower bounds follow. We also show that this implication is true even
if the simulation only holds for graph properties. Monotone simulations for graph properties are motivated by a result of Rossman \[ {\text{Ros08a}} \], which shows that very strong monotone simulations are possible for homomorphism-preserving graph properties. The lower bounds from monotone simulations are proved with the simulation result and padding argument used in the previous section (Lemmas 2.1 and 3.6).

4.1 A non-trivial simulation for bounded-depth circuits

The earliest monotone simulation result was proved for DNFs by Quine \[ {\text{Qui53}} \].

**Theorem 4.1** (Quine \[ {\text{Qui53}} \]). For all \( s : \mathbb{N} \rightarrow \mathbb{N} \), we have \( \text{DNF}[s] \cap \text{Mono} \subseteq \text{mDNF}[s] \).

**Proof.** If a given DNF computes a monotone Boolean function, simply removing the negative literals continues to compute the same function. Let \( \text{DT}_{\text{size}}(f) \) denote the size of a smallest decision-tree computing \( f \). We will need a result obtained by Rossman \[ {\text{Ros17a}} \].

**Theorem 4.2** (\[ {\text{Ros17a}} \]). If \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) is computable by an \( \text{AC}^0 \) circuit of depth \( d \) and size \( s \), then \( \text{DT}_{\text{size}}(f) = 2^{(1-1/O(\log s)^{d-1})n} \).

**Theorem 4.3.** Let \( s : \mathbb{N} \rightarrow \mathbb{N} \) and \( d \geq 1 \). We have \( \text{AC}^0_d[s] \cap \text{Mono} \subseteq \text{mSIZE}[t] \), where \( t = n \cdot 2^{n(1-1/O(\log s)^{d-1})} \). Moreover, this upper bound is achieved by monotone DNFs of size \( t/n \).

**Proof.** Let \( f \) be a monotone function computable by an \( \text{AC}^0 \) circuit of depth \( d \) and size \( s \). By Theorem 4.2, there exists a decision tree of size \( 2^{(1-1/O(\log s)^{d-1})n} \) computing \( f \). Therefore, there exists a DNF of the same size computing \( f \), which can be taken to be monotone by Theorem 4.1. This can be converted into a monotone circuit of size \( n \cdot 2^{n(1-1/O(\log s)^{d-1})} \).

We observe that it is possible to immediately deduce an \( \text{AC}^0 \) lower bound for Majority using this simulation theorem. Even though near-optimal lower bounds for Majority have been known for a long time \[ {\text{Hås86}} \] and the proof of the main technical tool (Theorem 4.2) behind our simulation result is similar to the one used by \[ {\text{Hås86}} \], the argument below illustrates how a monotone simulation can lead to non-monotone circuit lower bounds.

**Corollary 4.4.** Any depth-\( d \) \( \text{AC}^0 \) circuit computing Majority has size \( 2^{\Omega((n/\log n)^{1/(d-1)})} \).

**Proof.** Note that Majority has \( \binom{n}{n/2} = \Omega(2^n/\sqrt{n}) \) minterms. Therefore, any monotone DNF computing Majority has size at least \( \Omega(2^n/\sqrt{n}) \). By Theorem 4.3, it follows that the size \( s \) of a depth-\( d \) \( \text{AC}^0 \) computing Majority satisfies the following inequality:

\[
2^{n(1-1/O(\log s)^{d-1})} = \Omega(2^{n-\frac{1}{2}\log n}).
\]

From this equation we obtain \( s = 2^{\Omega((n/\log n)^{1/(d-1)})} \). 

4.2 Non-monotone lower bounds from monotone simulations

We now show that if monotone circuits are able to efficiently simulate non-monotone circuits computing monotone Boolean functions, then striking complexity separations follow. We also show a result of this kind for simulations of graph properties. We first prove a lemma connecting the simulation of $AC^0$ circuits with the simulation of $NL$ machines.

Lemma 4.5. For all constants $\varepsilon > 0$ and $C \geq 1$, if $AC^0 \cap Mono \subseteq mSIZE[2^{O((\log n)^C)}]$, then $NL \cap Mono \subseteq mSIZE[2^{o(n^\varepsilon)}]$.

Proof. We prove the contrapositive. Suppose that there exists $\varepsilon > 0$ such that $NL \cap Mono \not\subseteq mSIZE[2^{o(n^\varepsilon)}]$. This means that there exists a monotone function $f$ such that $f \in NL$ and any monotone circuit computing $f$ has size $2^{\Omega(n^\varepsilon)}$.

Let $\delta = \varepsilon/(2C)$ and let $m = 2^{n^\delta}$. Let $g : \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}$ be the Boolean function defined as $g(x,y) = f(x)$. Note that $g$ is a function on $N := m + n = 2^{\Theta(n^\delta)}$ bits. By Lemma 2.1, there exists an $AC^0$ circuit computing $f$ of size $2^{n^{\delta}} = 2^{O(\log N^{1/\delta})} = 2^{\Omega((\log n)^{2C})}$.

Next, we recall the strongest known monotone circuit and formula lower bounds for a monotone function in $NP$.

Theorem 4.6 ([PR17]). $NP \cap Mono \not\subseteq mDEPTH[o(n)]$.

Theorem 4.7 ([CKR20]). $NP \cap Mono \not\subseteq mSIZE[2^{o(\sqrt{n}/\log n)}]$.

We are now ready to state and prove our first result regarding new complexity separations from monotone simulations. Recall that obtaining explicit lower bounds against depth-3 $AC^0$ circuits of size $2^{\omega(n^{1/2})}$ is a major challenge in circuit complexity theory, while the best lower bound on the size of depth-4 $AC^0$ circuits computing a function in $NP$ is currently $2^{\Omega(n^{1/3})}$ [Hås86]. Moreover, no strict separation is known in the following sequence of inclusions of complexity classes: $ACC \subseteq TC^0 \subseteq NC^1 \subseteq L \subseteq NL \subseteq \oplus L \subseteq NC^2$. We show that efficient monotone simulations would bring new results in both of these fronts. (We stress that all lower bound consequences appearing below refer to separations against non-uniform circuits.)

Theorem 4.8. Let $C$ be a class of circuits. There exists $\varepsilon > 0$ such that the following holds:

1. If $AC^0_3 \cap Mono \subseteq mNC^1$, then $NP \not\subseteq AC^0_3[2^{o(n)}]$.

2. If $AC^0_1 \cap Mono \subseteq mSIZE[poly]$, then $NP \not\subseteq AC^0_4[2^{o(\sqrt{n}/\log n)}]$.

3. If $C \cap Mono \subseteq mSIZE[2^{O(n^\varepsilon)}]$, then $NC^2 \not\subseteq C$.

4. If $AC^0 \cap Mono \subseteq mSIZE[poly]$, then $NC^2 \not\subseteq NC^1$.

Proof. We will prove each item separately.

---

14In other words, all upper bounds are uniform, but the lower bounds hold even for non-uniform circuits. Note that this is stronger than lower bounds for uniform circuits.
Proof of (1). Let us assume that $AC^0_3 \cap \text{Mono} \subseteq mNC^1$. Let $f$ be the function of Theorem 4.6. For a contradiction, suppose that $f \in AC^0_3[2^{o(n)}]$. Let $\alpha : N \to N$ be such that $\alpha(n) \to \infty$ and $f$ has a depth-3 $AC^0$ circuit of size $2^{n/\alpha}$. Let $m = 2^{n/(10\alpha)}$ and let $g : \{0, 1\}^n \times \{0, 1\}^m \to \{0, 1\}$ be the function $g(x, y) = f(x)$. Let $N = n + m = (1 + o(1))2^{n/(10\alpha)}$. Clearly, the function $g$ has a depth-3 $AC^0$ circuit of size $2^{n/\alpha} = N^{O(1)}$. Since $g$ is monotone, we conclude from the assumption that $g$ is computed by a polynomial-size monotone formula. Now, since $f(x) = g(x, 1^n)$, we obtain a monotone formula of size $N^{O(1)} = 2^{o(n)}$ for computing $f$, which contradicts the lower bound of Theorem 4.6.

Proof of (2). Similar to the proof of item (1), but using Theorem 4.7 instead.

Proof of (3). Suppose that $NC^2 \subseteq C$. By Theorem 3.1, there exists a monotone function $f \in NC^2$ on $n$ bits and a number $\epsilon > 0$ such that $f \notin mSIZE[2^{o(n^\delta)}]$. Therefore, for any $\delta > 0$ such that $\delta < \epsilon$, we have $f \notin mSIZE[2^{o(n^\delta)}]$. Since, by assumption, we have $f \in NC^2 \subseteq C$, we obtain $C \cap \text{Mono} \not\subseteq mSIZE[2^{o(n^\delta)}]$.

Proof of (4). If $NC^2 \subseteq NC^1$, then, by item (3), we get $NC^1 \cap \text{Mono} \not\subseteq mSIZE[2^{o(n^\delta)}]$. From Lemma 4.5, we obtain $AC^0 \cap \text{Mono} \not\subseteq mSIZE[\text{poly}]$. □

As a motivation to the ensuing discussion, we recall a result of Rossman [Ros08a], who showed that any homomorphism-preserving graph property computed by $AC^0$ circuits is also computed by monotone $AC^0$ circuits.

Theorem 4.9 ([Ros08a]). $AC^0 \cap \text{HomPreserving} \subseteq mDNF[\text{poly}]$.

This inspires the question of whether general graph properties can also be efficiently simulated by monotone circuits. We show that, if true, such simulations would imply strong complexity separations. Let us first recall an exponential monotone circuit lower bound for monotone graph properties, and we will be ready to state and prove our main result.

Theorem 4.10 ([AB87]). There exists $\epsilon > 0$ such that $NP \cap \text{Mono} \cap \text{GraphProperties} \not\subseteq mSIZE[2^{o(n^\delta)}]$.

Theorem 4.11. Let $C$ be a class of circuits. The following holds:

1. If $C \cap \text{Mono} \cap \text{GraphProperties} \subseteq mSIZE[\text{poly}]$, then $L \not\subseteq C$.

2. If $C \cap \text{Mono} \cap \text{GraphProperties} \subseteq mDEPTH[\text{poly}]$, where $n$ denotes the number of input bits, then $L \not\subseteq C$.

3. If $AC^0 \cap \text{Mono} \cap \text{GraphProperties} \subseteq mSIZE[\text{poly}]$, then $NP \not\subseteq NC^1$.

Proof. We will prove each item separately.

Proof of (1). Suppose that $L \subseteq C$. By Theorem 3.4, the monotone graph property OddFactor satisfies $\text{OddFactor} \notin mSIZE[\text{poly}]$. Moreover, we have the upper bound $\text{OddFactor} \in L$ by Theorem 3.5. Since, by assumption, we have $\text{OddFactor} \in L \subseteq C$, we obtain $C \cap \text{Mono} \cap \text{GraphProperties} \not\subseteq mSIZE[\text{poly}]$.

Proof of (2). Suppose that $L \subseteq C$. By Theorems 3.4 and 3.5, there exists a monotone graph property $f \in L$ such that $f \notin mDEPTH[\text{poly}]$. Since, by assumption, we have $f \in L \subseteq C$, we obtain $C \cap \text{Mono} \cap \text{GraphProperties} \not\subseteq mDEPTH[\text{poly}]$. □
Proof of (3). Suppose that $\text{NP} \subseteq \text{NC}^1$. By Theorem 4.10, there exists a monotone graph property $f \in \text{NC}^1$ such that $\text{mSIZE}(f) = 2\Omega(n^\varepsilon)$ for some $\varepsilon > 0$. Let $\delta = \varepsilon/2$. By Lemma 3.6 (Item 2), there exists a monotone graph property $g$ on $N = 2^{n^\delta}$ vertices computed by an $\text{AC}^0$ circuit of size $N^{2+o(1)}$ such that $f$ is a monotone projection of $g$. Theorem 4.10 implies that any monotone circuit computing $f$ has size $2^{\Omega(n^\varepsilon)} = 2^{\Omega((\log N)^2)} = N^{\omega(1)}$.

\[ \Box \]

5 Monotone Complexity of Constraint Satisfaction Problems

In this section, we study the monotone complexity of Boolean-valued CSPs. Our goal is to classify which types of Boolean CSPs are hard for monotone circuit size and monotone circuit depth, eventually proving Theorems 1.4 and 1.5.

We will first spend some time recalling standard definitions and concepts in the theory of CSPs (Section 5.1), as well as a few results about CSPs that were proved in previous works [Sch78, Jea98, BCRV03, BCRV04, ABI+09] (Section 5.2). We will then prove Theorem 1.5 in Section 5.3, and we will finally prove Theorem 1.4 in Section 5.5 after proving some auxiliary results in Section 5.4.

5.1 Definitions

For a good introduction to the concepts defined below, we refer the reader to [BCRV03, BCRV04]. We also refer the reader to Section 1.1.3 for the definition of the family of functions $\text{CSP-SAT}_S$, as well as the terms constraint application, $S$-formula and satisfiable formula.

We denote by $p^n_i : \{0,1\}^n \rightarrow \{0,1\}$ the $i$-th projection function on $n$ variables, whose operation is defined as $p^n_i(x) = x_i$. For a set of Boolean functions $B$, we denote by $[B]$ the closure of $B$, defined as follows: a Boolean function $f$ is in $[B]$ if and only if $f \in B \cup \{\text{Identity}\}$ or if there exists $g \in B$ and $h_1, \ldots, h_k$ such that $f = g(h_1, \ldots, h_k)$, where each $h_i$ is either a projection function or a function from $[B]$. We can equivalently define $[B]$ as the set of all Boolean functions that can be computed by circuits using the functions of $B$ as gates. Note that $[B]$ necessarily contains an infinite number of Boolean functions, since $p^n_i \in [B]$ for every $n \in \mathbb{N}$; moreover, the constant functions are not necessarily in $[B]$. We say that $B$ is a clone if $B = [B]$. A few prominent examples of clones are the set of all Boolean functions (equal to $\{(\land, -)\}$), monotone functions (equal to $\{(\land, \lor, 0, 1)\}$), and linear functions (equal to $\{(\oplus, 1)\}$).

Remark 5.1. The set of all clones forms a lattice, known as Post’s lattice, under the operations $[A] \cap [B] := [A] \cap [B]$ and $[A] \cup [B] := [A \cup B]$. From the next section onwards, we will refer to the clones defined in [BCRV03] (such as $I_0, I_1$, etc.), assuming the reader is familiar with them. For the unfamiliar reader, we refer to Appendix C and Figures 1 and 4, which contain all the definitions of the clones we will need, as well as the entire Post’s lattice in graphical representation.

To avoid confusion, we will always refer to clones with normal-Roman font (e.g., $S_1, I_0$, etc.).

Let $S$ be a finite set of Boolean relations. We denote by $\text{CNF}(S)$ the set of all $S$-formulas. We denote by $\text{COQ}(S)$ the set of all relations which can be expressed with the following type of formula $\varphi$:

$$\varphi(x_1, \ldots, x_k) = \exists y_1, \ldots, y_{\ell} \psi(x_1, \ldots, x_k, y_1, \ldots, y_{\ell}),$$

where $\psi \in \text{CNF}(S)$. The relations in $\text{COQ}(S)$ will also be referred as conjunctive queries over $S$. We denote by $\langle S \rangle$ the set of relations defined as $\langle S \rangle := \text{COQ}(S \cup \{=\})$. If $S = \langle S \rangle$, we say that $S$
is a co-clone. We define

$$\text{CSP} = \{\text{CSP-SAT}_S : S \text{ is a finite set of relations} \}.$$  

We say that CSP-SAT$_S$ is trivial if CSP-SAT$_S$ is a constant function.

Let $R$ be a $k$-ary Boolean relation and let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function. For $x \in R$ and $i \in [k]$, we denote by $x[i]$ the $i$-th bit of $x$.

**Definition 5.2.** We say that $f$ is a polymorphism of $R$, and $R$ is an invariant of $f$, if, for all $x_1, \ldots, x_\ell \in R$, we have

$$(f(x_1[1], \ldots, x_\ell[1]), f(x_2[2], \ldots, x_\ell[2]), \ldots, f(x_k[k], \ldots, x_\ell[k])) \in R.$$  

We denote the set of all polymorphisms of $R$ by $\text{Pol}(R)$. For a set of relations $S$, we denote by $\text{Pol}(S)$ the set of all Boolean functions which are polymorphisms of all the relations of $S$. For a set of Boolean functions, we denote by $\text{Inv}(B)$ the set of all Boolean relations which are invariant under all functions of $B$ (i.e., $\text{Inv}(B) = \{R : R \subseteq \text{Pol}(R)\}$).

The following summarises the important facts about clones, co-clones and polymorphisms that are relevant to the study of CSPs [JCG97].

**Lemma 5.3.** Let $S$ and $S'$ be sets of Boolean relations and let $B$ and $B'$ be sets of Boolean functions. We have

(i) $\text{Pol}(S)$ is a clone and $\text{Inv}(B)$ is a co-clone;

(ii) If $S \subseteq S'$, then $\text{Pol}(S') \subseteq \text{Pol}(S)$;

(iii) If $B \subseteq B'$, then $\text{Inv}(B') \subseteq \text{Inv}(B)$;

(iv) $\text{COQ}(\text{COQ}(S)) = \text{COQ}(S)$;

(v) If $S \subseteq S'$, then $\text{COQ}(S) \subseteq \text{COQ}(S')$;

(vi) $\text{Inv}(\text{Pol}(S)) = \langle S \rangle$;

(vii) $\text{Pol}(\text{Inv}(B)) = [B]$.

We now define different types of reductions. We say that a reduction is a monotone OR-reduction if every bit of the reduction is either constant or can be computed by a monotone disjunction on the input variables. We write $f \leq^\text{mOR}_m g$ if there exists a many-one monotone OR-reduction from $f$ to $g$. We also write $f \leq^\text{AC}_m g$ if there exists a many-one AC$^0$ reduction from $f$ to $g$, and $f \leq^\text{mNL}_m g$ if there exists a many-one mNL reduction from $f$ to $g$\footnote{A many-one AC$^0$ (resp. mNL) reduction is one in which each bit of the reduction is either constant or can be computed with a polynomial-size AC$^0$ circuit (resp. monotone nondeterministic branching program). Recall that a monotone nondeterministic branching program is a directed acyclic graph $G$ with two distinguished vertices $s$ and $t$, in which each edge $e$ is labelled with an input function $\rho_e \in \{0,1\}$. Given an input $x$, the program accepts if there exists a path from $s$ to $t$ in the subgraph $G_x$ of $G$ in which an edge $e$ appears if $\rho_e(x) = 1$.}. Unless otherwise specified, every reduction we consider will generate an instance of polynomial size on the length of the input.

Finally, we denote by $\text{OR}^k$ and $\text{NAND}^k$ the $k$-ary OR and NAND relations, respectively.
5.2 Basic facts about CSP-SAT

We state here basic facts about the CSP-SAT function. These facts are proved in the original paper of Schaefer [Sch78], as well as in later papers [Jea98, BCRV03, BCRV04, ABI+09].

Lemma 5.4 below is one of the most important lemmas of this section and will be used many times. It states that Pol(S) characterises the monotone complexity of CSP-SAT_S, in the sense that the sets of relations with few polymorphisms give rise to the hardest instances of CSPs. A non-monotone version of this result was proved in [Jea98, BCRV04, Theorem 2.4], and we check in Appendix B.2 that their proofs also hold in the monotone case.

Lemma 5.4 (Polymorphisms characterise the complexity of CSPs [Jea98, BCRV04, Theorem 2.4]). If Pol(S_2) ⊆ Pol(S_1), then CSP-SAT^n_{S_1} ≤^{NL}_{m} CSP-SAT^{poly(n)}_{S_2}.

Theorem 5.5 gives monotone circuit upper bounds for some instances of CSP-SAT_S. Non-monotone variants of this upper bound were originally obtained in the seminal paper of Schaefer [Sch78], and we again check that the monotone variants work in Appendix B.3.

Theorem 5.5 (Monotone version of the upper bounds for CSP-SAT [Sch78, ABI+09]). Let S be a finite set of relations. The following holds.

1. If E_2 ⊆ Pol(S) or V_2 ⊆ Pol(S), then CSP-SAT_S ∈ mSIZE[poly].
2. If D_2 ⊆ Pol(S), or S_{00} ⊆ Pol(S), or S_{10} ⊆ Pol(S), then CSP-SAT_S ∈ mNL.

Finally, we state here a result of [ABI+09], which classifies the non-monotone complexity of CSP-SAT_S under ≤^AC^0 reductions. The classification of the complexity of CSP-SAT_S is based solely on Pol(S). See Figure 1 for a graphical representation.

Theorem 5.6 (Refined classification of CSP problems [ABI+09, Theorem 3.1]). Let S be a finite set of Boolean relations. The following holds.

- If I_0 ⊆ Pol(S) or I_1 ⊆ Pol(S), then CSP-SAT_S is trivial.
- If Pol(S) ∈ \{I_2, N_2\}, then CSP-SAT_S is ≤_{m}^{AC^0}-complete for NP.
- If Pol(S) ∈ \{V_2, E_2\}, then CSP-SAT_S is ≤_{m}^{AC^0}-complete for P.
- If Pol(S) ∈ \{L_2, L_3\}, then CSP-SAT_S is ≤_{m}^{AC^0}-complete for ⊕L.
- If S_{00} ⊆ Pol(S) ⊆ S_{00}^2 or S_{10} ⊆ Pol(S) ⊆ S_{10}^2 or Pol(S) ∈ \{D_2, M_2\}, then CSP-SAT_S is ≤_{m}^{AC^0}-complete for NL.
- If Pol(S) ∈ \{D_1, D\}, then CSP-SAT_S is ≤_{m}^{AC^0}-complete for L.
- If S_{02} ⊆ Pol(S) ⊆ R_2 or S_{12} ⊆ Pol(S) ⊆ R_2, then either CSP-SAT_S ∈ AC^0 or CSP-SAT_S is ≤_{m}^{AC^0}-complete for L.

5.3 A monotone dichotomy for CSP-SAT

In this section, we prove Theorem 1.5. We first prove Part (1) of the theorem (the dichotomy for circuit size), and then we prove Part (2) of the theorem (the dichotomy for circuit depth).
Figure 1: Graph of all closed classes of Boolean functions. The vertices are colored with the complexity of deciding CSPs whose set of polymorphisms corresponds to the label of the vertex. Trivial CSPs are those that correspond to constant functions. Every hardness result is proved under \( \leq_{AC^0} \) reductions. See Theorem 5.6 for details. A similar figure appears in [ABI+09, Figure 1].
Dichotomy for circuits. To prove the dichotomy for circuits, we first show that, for any set of relations \( S \) whose set of polymorphisms is contained in \( L_3 \), we can monotonically reduce 3-XOR-SAT to CSP-SAT\(_S\).

**Lemma 5.7.** Let \( S \) be a finite set of relations. If \( \text{Pol}(S) \subseteq L_3 \), then 3-XOR-SAT \( \leq_{mNL} \) CSP-SAT\(_S\).

**Proof.** Inspecting Post’s lattice (Figure 1), note that the only clones strictly contained in \( L_3 \) are \( L_2, N_2 \) and \( I_2 \). We will first show that the reduction holds for the case \( \text{Pol}(S) = L_2 \) and then prove that the reduction also holds for the case \( \text{Pol}(S) = L_3 \). Lemma 5.4 will then imply the cases \( \text{Pol}(S) \in \{N_2, I_2\} \), since \( I_2 \subseteq N_2 \subseteq L_3 \).

It’s not hard to check that, if \( \text{Pol}(S) = L_2 \), then \( \text{Pol}(S) \subseteq \text{Pol}(3\text{-XOR-SAT}) \) (it suffices to observe that bitwise XORing three satisfying assignments to a linear equation gives rise to a new satisfying assignment to the same equation). Therefore, from Lemma 5.4 we deduce that 3-XOR-SAT admits a reduction to CSP-SAT\(_S\) in \( mNL \). In order to prove the case \( \text{Pol}(S) = L_3 \), we first prove the following claim.

**Claim** ([ABI+09, Lemma 3.11]). Let \( S \) be a finite set of relations such that \( \text{Pol}(S) = L_2 \). There exists a finite set of relations \( S' \) such that \( \text{Pol}(S') = L_3 \) and CSP-SAT\(_S^{S'} \) \( \leq_{mProj} \) CSP-SAT\(_{S'}^{n+1} \).

**Proof.** We describe the proof of Lemma 3.11 in [ABI+09] and observe that it gives a monotone reduction.

For a relation \( R \in S \), let \( R' = \{ (\neg x_1, \ldots, \neg x_k) : (x_1, \ldots, x_k) \in R \} \). Let also \( S' = \{ R' : R \in S \} \). It’s not hard to check that \( \text{Pol}(S') = L_3 \), since \( S' \) is an invariant of \( L_2 \) and \( N_2 \), and \( L_3 \) is the smallest clone containing both \( L_2 \) and \( N_2 \); moreover, if \( \rho \in \text{Pol}(S') \) and \( \rho \) is a Boolean function on at least two bits, then \( \rho \in \text{Pol}(S) = L_2 \).

Now let \( F \) be a instance of CSP-SAT\(_S^{n} \). For every constraint \( C = R(x_1, \ldots, x_k) \) in \( F \), we add the constraint \( C' = R'(\alpha, x_1, \ldots, x_k) \) to the \( S' \)-formula \( F' \), where \( \alpha \) is a new variable. Note that \( F' \) is a \( S' \)-formula, defined on \( n + 1 \) variables, which is satisfiable if and only if \( F \) is satisfiable. Moreover, the construction of \( F' \) from \( F \) can be done with a monotone projection. \( \square \)

Since the case \( \text{Pol}(S) = L_2 \) holds, the case \( \text{Pol}(S) = L_3 \) now follows from Lemma 5.4 and the Claim. Finally, from Lemma 5.4 we conclude that the reduction also holds for the case \( \text{Pol}(S) \in \{N_2, I_2\} \), since \( I_2 \subseteq N_2 \subseteq L_3 \). \( \square \)

**Theorem 5.8** (Dichotomy for monotone circuits). Let \( S \) be a finite set of relations. If \( \text{Pol}(S) \subseteq L_3 \) then there exists a constant \( \varepsilon > 0 \) such that \( \text{mSIZE}(\text{CSP-SAT}_S) = 2^{\Omega(n^\varepsilon)} \). Otherwise, we have \( \text{mSIZE}(\text{CSP-SAT}_S) = n^{O(1)} \).

**Proof.** If \( \text{Pol}(S) \subseteq L_3 \), the lower bound follows from the ‘moreover’ part of Theorem 3.1, and Lemma 5.7. For the upper bound, we inspect Post’s lattice (Figure 1). Observe that, if \( \text{Pol}(S) \not\subseteq L_3 \), the following are the only possible cases:

1. \( I_0 \subseteq \text{Pol}(S) \) or \( I_1 \subseteq \text{Pol}(S) \). In both cases, any CNF\((S)\) is trivially satisfiable.
2. \( E_2 \subseteq \text{Pol}(S) \) or \( V_2 \subseteq \text{Pol}(S) \). In this case, CSP-SAT\(_S \in \text{mSIZE}[\text{poly}] \) by Theorem 5.5.
3. \( D_2 \subseteq \text{Pol}(S) \). In this case, CSP-SAT\(_S \in \text{mNL} \subseteq \text{mSIZE}[\text{poly}] \) by Theorem 5.5. \( \square \)
Solvable in mSIZE[poly].
Requires monotone circuits of size $2^{\Omega(n^\epsilon)}$.
Trivial (i.e., a constant function).

**Figure 2:** Illustration of Theorem 5.8. The vertices are colored with the monotone circuit size complexity of deciding CSPs whose set of polymorphisms corresponds to the label of the vertex.
**Remark 5.9.** We remark that the lifting theorem of [GKRS19] (which is an ingredient in the proof of Theorem 3.1) is only used to prove that the monotone complexity of CSP-SAT$_S$ is exponential when Pol($S$) $\subseteq$ L$_3$. If we only care to show a superpolynomial separation, then it suffices to apply the superpolynomial lower bound for CSPs with counting proved in [FV98, BGW99] using the approximation method. Indeed, we give an explicit proof in Appendix A. The same holds for the consequences of this theorem (see Theorem 5.19).

**Dichotomy for formulas.** Define 3-Horn-SAT$_n : \{0, 1\}^{2n^3+n} \rightarrow \{0, 1\}$ as 3-Horn-SAT$_n = \text{CSP-SAT}^n_{\mathcal{H}^3}$, where

$$\mathcal{H}^3 = \{(-x_1 \lor -x_2 \lor x_3), (-x_1 \lor -x_2 \lor -x_3), (x)\}.$$ 

The following is proved in [RM99, GKRS19].

**Theorem 5.10 ([RM99, GKRS19]).** There exists $\varepsilon > 0$ such that 3-Horn-SAT $\in$ mSIZE[pol] \ mDEPTH[o(n$^\varepsilon$)].

**Proof sketch.** Since $E_2 \subseteq \text{Pol}(\mathcal{H}^3)$ (see, e.g., [CKS01, Lemma 4.8]), the upper bound follows from Theorem 5.5. The lower bound follows from a lifting theorem of [RM99, GKRS19]. They show that the monotone circuit-depth of 3-Horn-SAT is at least the depth of the smallest Resolution-tree refuting a so-called pebbling formula. Since this formula requires Resolution-trees of depth $n^\varepsilon$, the lower bound follows.

Analogously to the previous section, we show that 3-Horn-SAT reduces to CSP-SAT$_S$ whenever Pol($S$) is small enough, in a precise sense stated below. We then deduce the dichotomy for formulas with a similar argument.

**Lemma 5.11.** Let $S$ be a finite set of relations. If Pol($S$) $\subseteq$ E$_2$ or Pol($S$) $\subseteq$ V$_2$, then 3-Horn-SAT $\leq$ mNL$_m$ CSP-SAT$_S$.

**Proof.** We first consider the case Pol($S$) $\subseteq$ E$_2$. Note that E$_2 \subseteq \text{Pol}(3\text{-Horn-SAT})$ (see, e.g., [CKS01, Lemma 4.8]). Therefore, from Lemma 5.4 we deduce that 3-Horn-SAT admits a reduction to CSP-SAT$_S$ in mNL.

Now let 3-AntiHorn-SAT = CSP-SAT$_{\mathcal{A}^3}$, where $\mathcal{A}^3 = \{(x_1 \lor x_2 \lor -x_3), (x_1 \lor x_2 \lor x_3), (-x)\}$. Observe that a $\mathcal{H}^3$-formula $\varphi$ is satisfiable if and only if the $\mathcal{A}^3$-formula $\varphi(-x_1, \ldots, -x_n)$ is satisfiable. Therefore, we have 3-Horn-SAT $\leq$ mProj$_m$ 3-AntiHorn-SAT. Observing that $V_2 \subseteq \text{Pol}(\mathcal{A}^3)$ (again, see e.g. [CKS01, Lemma 4.8]), the result now follows from Lemma 5.4 and the previous paragraph.

**Theorem 5.12** (Dichotomy for monotone formulas). Let $S$ be a finite set of relations. If Pol($S$) $\subseteq$ L$_3$, or Pol($S$) $\subseteq$ V$_2$, or Pol($S$) $\subseteq$ E$_2$, then there is a constant $\varepsilon > 0$ such that mDEPTH(CSP-SAT$_S$) = $\Omega(n^\varepsilon)$. Otherwise, we have CSP-SAT$_S$ $\in$ mNL $\subseteq$ mNC$_2$ $\subseteq$ mDEPTH[log$^2$ n].

**Proof.** We will first prove the lower bound. If Pol($S$) $\subseteq$ L$_3$, the lower bound follows from Theorem 5.8. If Pol($S$) $\subseteq$ V$_2$ or Pol($S$) $\subseteq$ E$_2$, the lower bound follows from Theorem 5.10 and Lemma 5.11.

By inspecting Post’s lattice (Remark 5.1), we see that the remaining cases are:

1. I$_0$ $\subseteq$ Pol($S$) or I$_1$ $\subseteq$ Pol($S$). In both cases, any CNF($S$) is trivially satisfiable.

2. S$_{00}$ $\subseteq$ Pol($S$), or S$_{10}$ $\subseteq$ Pol($S$), or D$_2$ $\subseteq$ Pol($S$). In all of those three cases, we have CSP-SAT$_S$ $\in$ mNL by Theorem 5.5.
Theorem 5.12: Solvable in $\text{mNL} \subseteq \text{mDEPTH}[O(\log^2 n)]$. Requires monotone depth $\Omega(n^\epsilon)$. Trivial (i.e., a constant function).

Figure 3: Illustration of Theorem 5.12. The vertices are colored with the monotone circuit depth complexity of deciding CSPs whose set of polymorphisms corresponds to the label of the vertex.
5.4 Some auxiliary results

In this section, we prove auxiliary results needed in the proof of a more general form of Theorem 1.4. In particular, we will prove that all CSP-SAT$_S$ which are in AC$^0$ are also contained in mAC$^0 \subseteq$ mNC$^1$. Moreover, we show that, if CSP-SAT$_S \notin$ mNC$^1$, then CSP-SAT$_S$ is L-hard under \( \leq_{m^{AC^0}} \) reductions.

We first observe that, when COQ(S$_1$) \( \subseteq \) COQ(S$_2$), there exists an efficient low-depth reduction from CSP-SAT$_S$$_1$ to CSP-SAT$_S$$_2$. This reduction, which will be useful in this section, is more refined than the one given by Lemma 5.4. A proof of the non-monotone version of this statement is found in [BCRV04, Proposition 2.3], and we give a monotone version of this proof in Appendix B.2.

Lemma 5.13 ([BCRV04, Proposition 2.3]). If COQ(S$_1$) \( \subseteq \) COQ(S$_2$), then there exists a constant \( C \in \mathbb{N} \) such that CSP-SAT$_{S_1}^n \leq_{m^{OR}}$ CSP-SAT$_{S_2}^n$.

Proof. We defer the proof to Appendix B.2.

We now recall some lemmas from [ABI+09], and prove a few consequences from them. We say that a set \( S \) of relations can express equality if \( \{=\} \subseteq \text{COQ}(S) \).

Lemma 5.14 ([ABI+09]). Let \( S \) be a finite set of relations. Suppose \( S_{02} \subseteq \text{Pol}(S) \) (\( S_{12} \subseteq \text{Pol}(S) \), resp.) and that \( S \) cannot express equality. Then there exists \( k \geq 2 \) such that \( S \subseteq \text{COQ}(\{\text{OR}^k, x, \neg x\}) \) (\( S \subseteq \text{COQ}(\{\text{NAND}^k, x, \neg x\}) \), resp.).

Proof. Follows from the proof of Lemma 3.8 of [ABI+09].

Lemma 5.15. Let \( S \) be a finite set of relations such that \( \text{Pol}(S) \subseteq R_2 \). If \( S_{02} \subseteq \text{Pol}(S) \) or \( S_{12} \subseteq \text{Pol}(S) \), and \( S \) cannot express equality, then CSP-SAT$_S \in$ mAC$^0$.

Proof. We write the proof in the case \( S_{02} \subseteq \text{Pol}(S) \). The other case is analogous. From Lemmas 5.13 and 5.14 and Items (iv) and (v) of Lemma 5.3, we get that there is a monotone OR-reduction from CSP-SAT$_S$ to CSP-SAT$_{\{\text{OR}^k, x, \neg x\}}$ for some \( k \). However, an \( \{\text{OR}^k, x, \neg x\} \)-formula is unsatisfiable iff there exists a literal and its negation as a constraint in the formula, or if there exists a disjunction in the formula such that every one of its literals appears negatively as a constraint. This condition can be easily checked by a polynomial-size monotone DNF. Composing the monotone DNF with the monotone OR-reduction, we obtain a depth-3 AC$^0$ circuit computing CSP-SAT$_S$.

Lemma 5.16 ([ABI+09, Lemma 3.8]). Let \( S \) be a finite set of relations such that \( \text{Pol}(S) \subseteq R_2 \). If \( S_{02} \subseteq \text{Pol}(S) \) or \( S_{12} \subseteq \text{Pol}(S) \), and \( S \) can express equality, then CSP-SAT$_S$ is L-hard under \( \leq_{m^{AC^0}} \) reductions.

Lemma 5.17. Let \( S \) be a finite set of relations. If \( S_{02} \not\subseteq \text{Pol}(S) \) and \( S_{12} \not\subseteq \text{Pol}(S) \), then CSP-SAT$_S$ is L-hard or trivial.

Proof. This follows by inspecting Post’s lattice (Figure 1) and the classification theorem (Theorem 5.6).

We may now prove the main result of this subsection.

Theorem 5.18. We have CSP \( \cap \) AC$^0$ \( \subseteq \) mAC$^0$$_3$. Moreover, if CSP-SAT$_S$ \( \notin \) mAC$^0$$_3$, then CSP-SAT$_S$ is L-hard under \( \leq_{m^{AC^0}} \) reductions.
Proof. Let $S$ be a finite set of relations. If $\text{CSP-SAT}_S \not\in \text{mAC}^0_3$, then, by Lemma 5.15, at least one of the following cases hold:

1. $S_{02} \subseteq \text{Pol}(S) \subseteq R_2$ or $S_{12} \subseteq \text{Pol}(S) \subseteq R_2$, and $S$ can express the equality relation;

2. $S_{02} \nsubseteq \text{Pol}(S) \subseteq R_2$ and $S_{12} \nsubseteq \text{Pol}(S) \subseteq R_2$.

3. $\text{Pol}(S) \nsubseteq R_2$.

Since CSP-SAT$_S$ is not trivial, we obtain that CSP-SAT$_S$ is L-hard in the first two cases by Lemmas 5.16 and 5.17, and it’s easy to check that CSP-SAT$_S$ is also L-hard in the third case by inspecting Post’s lattice (Figure 1) and the classification theorem (Theorem 5.6). Since L $\not\subseteq$ AC$^0$, this also implies that, if CSP-SAT$_S$ $\in$ AC$^0$, then $S_{02} \subseteq \text{Pol}(S) \subseteq R_2$ or $S_{12} \subseteq \text{Pol}(S) \subseteq R_2$, and $S$ cannot express the equality relation. Lemma 5.15 again gives CSP-SAT$_S$ $\in$ mAC$^0_3$. □

5.5 Consequences for monotone circuit lower bounds via lifting

We now prove a stronger form of Theorem 1.4. In the previous section, we showed that CSP $\cap$ AC$^0$ $\subseteq$ mAC$^0$. In particular, this means that there does not exist a finite set of relations $S$ such that CSP-SAT$_S$ separates AC$^0$ and mNC$^1$, a separation which we proved in Theorem 1.1. We will also observe that, if CSP-SAT$_S$ $\not\in$ mNC$^2$, then CSP-SAT$_S$ is $\oplus$L-hard.

Theorem 5.19. Let $S$ be a finite set of Boolean relations.

1. If CSP-SAT$_S$ $\not\in$ mAC$^0_3$ then CSP-SAT$_S$ is L-hard under $\leq_m^{AC^0}$ reductions.

2. If CSP-SAT$_S$ $\not\in$ mNC$^2$, then CSP-SAT$_S$ is $\oplus$L-hard under $\leq_m^{AC^0}$ reductions.

Proof. Item (1) follows from Theorem 5.18. To prove item (2), suppose that mDEPTH(CSP-SAT$_S$) $= \omega(\log^2 n)$. Then, by Theorem 5.12, we conclude that Pol$(S) \subseteq L_3$, or Pol$(S) \subseteq V_2$, or Pol$(S) \subseteq E_2$. Theorem 5.6 implies that CSP-SAT$_S$ is $\oplus$L-hard. □

Further Discussion. We recall the discussion of Section 1.1.3. We introduced and defined the functions CSP-SAT$_S$ in that section, as a way to capture monotone circuit lower bounds proved via lifting. This in particular captures the monotone function 3-XOR-SAT, which was proved in [GKRS19] to require monotone circuit lower bounds of size $2^{\Omega(1)}$ to compute, even though $\oplus$L-machines running in polynomial-time can compute it. Theorem 5.19 proves that this separation between monotone and non-monotone circuit lower bounds cannot be improved by varying the set of relations $S$, as we argue below.

There are two ways one could try to find a function in AC$^0$ with large monotone complexity using a CSP-SAT function. First, one could try to define a set of relations $S$ such that CSP-SAT$_S$ $\in$ AC$^0$, but the monotone complexity of CSP-SAT$_S$ is large. However, Item (1) of Theorem 5.19 proves that this is impossible, as any CSP-SAT function outside of mAC$^0$ is L-hard under simple reductions and, therefore, cannot be computed in AC$^0$.

Secondly, one could try to be apply the arguments of Section 3, consisting of a padding trick and a simulation theorem. When $S$ is the set of 3XOR relations, then indeed we obtain a function in AC$^0[6]$ with superpolynomial monotone circuit complexity, as proved in Theorem 3.1. However, Item (2) of Theorem 5.19 proves that this is best possible, as any CSP-SAT function which admits a superpolynomial monotone circuit lower bound must be $\oplus$L-hard and, therefore, at least as hard
as $3$-$XOR$-$SAT$ for non-monotone circuits. Item (2) also shows that even $CSP$-$SAT$ functions with a $\omega(\log^2 n)$ monotone depth lower bound must be $\oplus L$-hard, which suggests that the arguments of Section 3 applied to a $CSP$-$SAT$ function are not able to prove the separation of Theorem 3.3.

A caveat to these impossibility results is in order. First, we only study Boolean-valued CSPs here, though the framework of lifting can also be applied in the context of non-Boolean CSPs. It's not clear if non-Boolean CSPs exhibit the same dichotomies for monotone computation we proved in this section. We remark that Schaefer’s dichotomy for Boolean-valued CSPs [Sch78] has been extended to non-Boolean CSPs [Zhu17, Bul17].

Secondly, the instances of $CSP$-$SAT$ generated by lifting do not cover the entirety of the minterms and maxterms of $CSP$-$SAT$. In particular, our results do not rule out the possibility that a clever interpolation of the instances generated by lifting may give rise to a function that is easier to compute by non-monotone circuits, and therefore bypasses the hardness results of Theorem 5.19. One example is the Tardós function [Tar88]. A lifting theorem applied to a Pigeonhole Principle formula can be used to prove a lower bound on the size of monotone circuits that accept cliques of size $k$ and reject graphs that are $(k - 1)$-colorable, for some $k = n^\varepsilon$ [RM99, Rez23]. A natural interpolation for these instances would be the $k$-Clique function, which, being $NP$-complete, would be related to an $NP$-complete $CSP$-$SAT$. However, as proved by [Tar88], there is a monotone function in $P$ which has the same output behaviour over these instances.

References


32


Shuichi Hirahara. NP-hardness of learning programs and partial MCSP. In *Symposium on Foundations of Computer Science (FOCS), 2022*.


[Qui53] W. V. Quine. Two theorems about truth functions. Bol. Soc. Mat. Mexicana, 10:64–70, 1953. 4, 6, 11, 19


[Rez23] Susanna Rezende. Personal communication, 2023. 32


[Ros17a] Benjamin Rossman. An entropy proof of the switching lemma and tight bounds on the decision-tree size of $AC^0$. 2017. 18, 19


36
A Lower Bound for 3-XOR-SAT Using the Approximation Method

As discussed in Section 1.1.3, [GKRS19] obtained an exponential lower bound on the monotone circuit size of the function 3-XOR-SAT using techniques from communication complexity and lifting. Here we observe that a weaker but still super-polynomial lower bound can be proved using the approximation method.

First, we recall the function OddFactor$_n : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ of Section 3.2, which accepts a given graph if the graph contains an odd factor, which is a spanning subgraph in which the degree of every vertex is odd. For convenience, in this section we consider a weaker version of OddFactor, which takes as an input a bipartite graph with $n$ vertices on each part, and accepts if the graph contains an odd factor. Let Bipartite-OddFactor$_n : \{0,1\}^{n^2} \to \{0,1\}$ be this function. We remark that the lower bounds of Babai, Gáé and Wigderson [BGW99] for OddFactor (Theorem 3.4) also hold for Bipartite-OddFactor. The proof of the monotone circuit lower bound in particular is essentially Razborov’s lower bound for Matching via the approximation method [Raz85b].

**Theorem A.1** ([BGW99]). We have

$$\text{mSIZE}(\text{Bipartite-OddFactor}_n) = n^{\Omega(\log n)} \quad \text{and} \quad \text{mDEPTH}(\text{Bipartite-OddFactor}_n) = \Omega(n).$$
We can reduce Bipartite-OddFactor to 3-XOR-SAT by noting that computing Bipartite-OddFactor$_n(M)$ on a given matrix $M \in \{0, 1\}^{n^2}$ is computationally equivalent to deciding the satisfiability of the following $\mathbb{F}_2$ linear system over variables $\{x_{ij}\}$:

- For all $i \in [n]$: $\bigoplus_{k=1}^n x_{ik} = 1$;
- For all $j \in [n]$: $\bigoplus_{k=1}^n x_{kj} = 1$;
- For all $i, j \in [n]$ such that $M_{ij} = 0$: $x_{ij} = 0$.

We can then use a circuit for 3-XOR-SAT to solve this system by using a standard trick of introducing new variables to reduce the number of variables that appear in each equation, as done in the textbook reduction from SAT to 3-SAT. As the corresponding reductions turn out to be monotone, this implies monotone circuit and formula lower bounds for 3-XOR-SAT. We note that a somewhat similar argument (in the non-monotone setting) appears in Feder and Vardi [FV98, Theorem 30] regarding constraint satisfaction problems with the ability to count.

In order to formalise this argument, we will need the following definition and results.

**Definition A.2.** Let $f$ be a Boolean function. We define dual$(f) : x \mapsto \neg f(\neg x)$ as the dual of $f$.

**Lemma A.3.** Let $f$ be a monotone Boolean function. We have $m\text{SIZE}(f) = m\text{SIZE}($dual$(f))$ and $m\text{DEPTH}(f) = m\text{DEPTH}($dual$(f))$.

**Proof.** The idea is to push negations to the bottom and eliminate double negations at the input layer. In other words, applying De Morgan rules, we can convert any $\{\land, \lor\}$-circuit computing $f$ into a circuit computing dual$(f)$ by swapping $\land$-gates for $\lor$-gates, and vice-versa. Moreover, this transformation preserves the depth of the circuit. $\square$

We are ready to describe a monotone reduction from the function Bipartite-OddFactor$_n$ to 3-XOR-SAT, which implies the desired lower bounds.

**Theorem A.4.** There exists $\varepsilon > 0$ such that

\[ m\text{SIZE}(3\text{-XOR-SAT}) = n^{\Omega(\log n)} \quad \text{and} \quad m\text{DEPTH}(3\text{-XOR-SAT}) = \Omega(n^\varepsilon). \]

**Proof.** Recall that the value of the function Bipartite-OddFactor$_n(M)$ on a given matrix $M \in \{0, 1\}^{n^2}$ is equal to 1 if the following system is satisfiable:

- For all $i \in [n]$: $\bigoplus_{k=1}^n x_{ik} = 1$;
- For all $j \in [n]$: $\bigoplus_{k=1}^n x_{kj} = 1$;
- For all $i, j \in [n]$ such that $M_{ij} = 0$: $x_{ij} = 0$.

We introduce some extra variables to reduce the number of variables in each equation in the following way. For every $i \in [n]$, introduce variables $z_{i1}, \ldots, z_{i(n-1)}$ and the equations

\[
\begin{align*}
z_{i1} & = x_{i1} \oplus x_{i2}, \\
z_{i2} & = z_{i1} \oplus x_{i3}, \\
& \quad \ldots \\
z_{i,(n-1)} & = z_{i,(n-2)} \oplus x_{i,n}, \\
z_{i,(n-1)} & = 1.
\end{align*}
\]
Now note that these equations imply $z_{i,(n-2)} = \bigoplus_{k=1}^{n} x_{ik} = 1$. For each “column” equation $\bigoplus_{k=1}^{n} x_{kj} = 1$, we also add variables $w_j, \ldots, w_{j(n-1)}$ as above. In total, we add at most $2n^2$ variables and $2n^2$ equations. Therefore, there is a system of linear equations on $O(n^2)$ variables, where each constraint contains at most 3 variables, which is satisfiable if and only if $\text{Bipartite-OddFactor}_n(M) = 1$. Moreover, it is easy to see that the characteristic vector $\alpha$ of the set of equations of this system can be computed from $M$ by an anti-monotone projection, as we activate a constraint that depends on the input when $M_{ij} = 0$.

Now let $f = \text{dual}(3\text{-XOR-SAT})$ and $\beta = \neg \alpha$. Since, by definition, $3\text{-XOR-SAT}$ accepts unsatisfiable systems, we get $\text{Bipartite-OddFactor}_n(M) = \neg 3\text{-XOR-SAT}(\alpha) = f(\beta)$ and that $\beta$ is a monotone projection of $M$. Therefore, by Lemma A.3, we obtain

$$m\text{SIZE}(\text{Bipartite-OddFactor}_n) \leq m\text{SIZE}(3\text{-XOR-SAT})$$

and

$$m\text{DEPTH}(\text{Bipartite-OddFactor}_n) \leq m\text{DEPTH}(3\text{-XOR-SAT}).$$

\[ \square \]

## B Schaefer’s Theorem in Monotone Complexity

### B.1 Connectivity and generation functions

We recall the definitions of two prominent monotone Boolean functions that have efficient monotone circuits. Let $\text{ST-CONN} : \{0,1\}^{n^2} \to \{0,1\}$ be the function that outputs 1 on a given directed graph $G$ if there exists a path from 1 to $n$ in $G$. Let $\text{GEN} : \{0,1\}^{n^3} \to \{0,1\}$ be the Boolean function which receives a set $T$ of triples $(i,j,k) \in [n]^3$, and outputs 1 if $n \in S$, where $S \subseteq [n]$ is the set generated with the following rules:

- **Axiom**: $1 \in S$,
- **Generation**: If $i, j \in S$ and $(i,j,k) \in T$, then $k \in S$.

The following upper bounds are well-known and easy to prove.

**Theorem B.1** ([Juk12, Exercise 7.3], [RM99]). We have $\text{ST-CONN} \in \text{mNL}$ and $\text{GEN} \in \text{mSIZE}[\text{poly}]$.

### B.2 Proof of reduction lemmas

Here we present monotonised versions of the proofs of [BCRV04, Propositions 2.2 - 2.4], which give a simplified presentation of the results of [Sch78].

**Lemma 5.13** ([BCRV04, Proposition 2.3]). If $\text{COQ}(S_1) \subseteq \text{COQ}(S_2)$, then there exists a constant $C \in \mathbb{N}$ such that $\text{CSP-SAT}^n_{S_1} \leq \text{mOR CSP-SAT}^n_{S_2}$.

**Proof.** If $\text{COQ}(S_1) \subseteq \text{COQ}(S_2)$, then each relation of $S_1$ can be represented as a conjunctive query over $S_2$. Let $F_1$ be a $S_1$-formula. For each constraint $C_1$ of $F_1$, there exists a formula $\varphi(C_1)$ in $\text{CNF}(S_2)$ such that $C_1$ is a projection of $\varphi(C_1)$ (i.e., $C_1$ is a conjunctive query of $\varphi(C_1)$). However, note that $C_1$ is satisfiable if and only if $\varphi(C_1)$ is satisfiable. So we can replace the constraint $C_1$ by the set of constraints in $\varphi(C_1)$. Doing this for every constraint in $F_1$, we obtain an $S_2$-formula $F_2$ which is satisfiable iff $F_1$ is satisfiable.
Now note that, to decide if a given constraint application \(C\) of \(S_2\) is in the reduction, it suffices to check if there exists a \(S_1\)-constraint \(C_1\) in \(F_1\) such that \(C\) is in \(\varphi(C_1)\). Using non-uniformity, this can be easily done by an OR over the relevant input bits.

Finally, we observe that, since the arities of each relation in \(S_1\) and \(S_2\) are constant, we only add a constant number of variables for each constraint to represent \(S_1\)-formulas with conjunctive queries over \(S_2\)-formulas. 

\[\square\]

**Lemma B.2.** Let \(S\) be a set of Boolean relations. We have \(\text{CSP-SAT}_{S \cup \{\neq\}} \leq_{m} \text{CSP-SAT}_{S}\).

**Proof.** Let \(F\) be a \((S \cup \{\neq\})\)-formula on \(n\) variables given as an input. Remember that \(F\) is given as a Boolean vector \(\alpha\), where each bit of \(\alpha\) represents the presence of a constraint application on \(n\) variables from \(S \cup \{\neq\}\). We first build an undirected graph \(G\) with the variables \(x_1, \ldots, x_n\) as vertices, and we put an edge between \(x_i\) and \(x_j\) if the constraint \(x_i = x_j\) appears in \(F\). Note that \(G\) can be constructed by a monotone projection from \(F\).

Let \(R \in S\) and let \(C = R(x_1, \ldots, x_n)\) be a constraint application of \(R\). If \(C\) appears in \(F\), we add to the system every constraint of the form \(C' = R(y_1, \ldots, y_n)\) such that, for every \(i \in [n]\), there exists a path from \(x_i\) to \(y_i\) in the graph \(G\). In this case, we say that \(C\) *generates* \(C'\). Let \(F_2\) be the formula that contains all non-equality constraints of \(F\), and all the non-equality constraints generated by a constraint in \(F\). It’s not hard to see that \(F\) is satisfiable if and only if \(F_2\) is satisfiable, and therefore the reduction is correct.

Moreover, the reduction can be done in monotone \(\text{NL}\) using the monotone \(\text{NL}\) algorithm for \(\text{ST-CONN}\) (Theorem B.1). Indeed, there are at most \(n^k\) constraint applications of a given relation \(R\) of arity \(k\). Therefore, to decide if a constraint \(C' = R(y_1, \ldots, y_n)\) appears in \(F_2\), it suffices to check if there exists a constraint application of \(R\) in \(F\) which generates \(C'\). This can be checked with \(n^k\) calls to \(\text{ST-CONN}\). 

\[\square\]

**Lemma 5.4 (Polymorphisms characterise the complexity of CSPs [Jea98, BCRV04, Theorem 2.4]).** If \(\text{Pol}(S_2) \subseteq \text{Pol}(S_1)\), then \(\text{CSP-SAT}_{S_1} \leq_{m} \text{CSP-SAT}_{S_2}^{\text{poly}(n)}\).

**Proof.** If \(\text{Pol}(S_2) \subseteq \text{Pol}(S_1)\), then from Lemma 5.3 (Items (iii), (v) and (vi)) we obtain \(\text{COQ}(S_1) \subseteq \langle S_1 \rangle \subseteq \langle S_2 \rangle = \text{COQ}(S_2 \cup \{\neq\})\). Therefore, by Lemmas 5.13 and B.2 we can do the following chain of reductions in monotone \(\text{NL}\):

\[
\text{CSP-SAT}_{S_1} \leq_{m} \text{CSP-SAT}_{S_2} \leq_{m} \text{CSP-SAT}_{S_2} \leq_{m} \text{CSP-SAT}_{S_2}.
\]

\[\square\]

### B.3 Monotone circuit upper bounds

We restate and prove the theorem.

**Theorem 5.5 (Monotone version of the upper bounds for CSP-SAT [Sch78, ABI+09]).** Let \(S\) be a finite set of relations. The following holds.

1. If \(E_2 \subseteq \text{Pol}(S)\) or \(V_2 \subseteq \text{Pol}(S)\), then \(\text{CSP-SAT}_S \in \text{mSIZE}[\text{poly}]\).
2. If \(D_2 \subseteq \text{Pol}(S)\), or \(S_{00} \subseteq \text{Pol}(S)\), or \(S_{10} \subseteq \text{Pol}(S)\), then \(\text{CSP-SAT}_S \in \text{mNL}\).

**Proof.** We prove each case separately.

**Proof of (1).** We first observe that 3-Horn-SAT (see definition in Section 5.3, Dichotomy for formulas) can be solved by a reduction to \(\text{GEN} \in \text{mSIZE}[\text{poly}]\). Indeed, we interpret each constraint
of the form \((\neg x_i \lor \neg x_j \lor x_k)\) (which is equivalent to \(x_i \land x_j \implies x_k\)) as a triple \((i, j, k)\), and constraints of the form \(x_i\) as a triple \((0, 0, i)\), and 2.

Let \(S \subseteq \{0, 1, 2, \ldots, n\}\) be the set generated by these triples, applying the generation rules of \textsc{Gen}, using \(0 \in S\) as the axiom. It suffices to check that there exists some constraint of the form \(\neg x_i \lor \neg x_j \lor \neg x_k\), such that \(\{i, j, k\} \subseteq S\). This process can be done with polynomial-size monotone circuits, invoking \textsc{Gen}. Therefore, it follows from Theorem B.1 that \(3\text{-Horn-SAT} \in \text{mSIZE}[\text{poly}]\).

Moreover, we recall that, if \(E_2 \subseteq \text{Pol}(S)\), then \(S \subseteq \text{COQ}(\mathcal{H}^3)\) (in other words, every \(S\)-formula can be written as a set of 3-Horn equations) – see, e.g., [CKS01, Lemma 4.8]. Therefore, from Items (iv) and (v) of Lemma 5.3 and Lemma 5.13, we conclude that \(\text{CSP-SAT}_S \subseteq_{m^\text{OR}} 3\text{-Horn-SAT} \in \text{mSIZE}[\text{poly}]\).

Now recall that, if \(V_2 \subseteq \text{Pol}(S)\), then \(S \subseteq \text{COQ}(\mathcal{A}^3)\), where \(\mathcal{A}^3\) is the set of width-3 Anti-Horn relations (i.e., \(\mathcal{A}^3 = \{(x_1 \lor x_2 \lor \neg x_3), (x_1 \lor x_2 \lor x_3), (\neg x)\}\); see [CKS01, Lemma 4.8] for a proof of this observation). But note that an \(\mathcal{A}^3\)-formula \(\varphi\) is satisfiable if and only if the \(\mathcal{H}^3\)-formula \(\varphi(\neg x_1, \ldots, \neg x_n)\) is satisfiable. Therefore by Lemma 5.13 and Items (iv) and (v) of Lemma 5.3, we have \(\text{CSP-SAT}_S \subseteq_{m^\text{OR}} \text{CSP-SAT}_{\mathcal{A}^3} \subseteq_{m^\text{Proj}} 3\text{-Horn-SAT} \in \text{mSIZE}[\text{poly}]\).

**Proof of (2).** We first prove the case \(D_2 \subseteq \text{Pol}(S)\). Let \(2\text{-SAT} = \text{CSP-SAT}_\Gamma\), where \(\Gamma = \{(x_1 \lor x_2), (x_1 \lor \neg x_2), (\neg x_1 \lor \neg x_2)\}\). It’s easy to check that the standard reduction from 2-SAT to \text{ST-CONN} can be done in monotone \(\text{NL}\) (see [JLL76, Theorem 4]). Therefore, it follows from Theorem B.1 that \(2\text{-SAT} \in \text{mNL}\). Now, recall that, if \(D_2 \subseteq \text{Pol}(S)\), then \(S \subseteq \text{COQ}(\Gamma)\) (see, e.g., [CKS01, Lemma 4.9]). Therefore, from Lemma 5.13 and Items (iv) and (v) of Lemma 5.3, we conclude \(\text{CSP-SAT}_S \subseteq \text{mNL}\).

We now suppose that \(S_{00} \subseteq \text{Pol}(S)\). We check that the proof of [ABI+09, Lemma 3.4] gives a monotone circuit. If \(S_{00} \subseteq \text{Pol}(S)\), then there exists \(k \geq 2\) such that \(S_{00}^{k} \subseteq \text{Pol}(S)\) (that’s because there does not exist a finite set of relations \(S\) such that \(\text{Pol}(S) = S_{00}\)). Note that \(S_{00}^{k} = \text{Pol}(\Gamma)\), where \(\Gamma = \{\text{OR}^k, x, \neg x, \rightarrow, =\}\). We show below how to decide if a \(\Gamma\)-formula is unsatisfiable in monotone \(\text{NL}\). The result then follows from Lemma 5.4.

Let \(F\) be a given \(\Gamma\)-formula with \(n\) variables. We first construct a directed graph \(G\), with vertex set \(\{x_1, \ldots, x_n\}\), and with arcs \((x_i, x_j)\) if \(x_i \rightarrow x_j\) is a constraint of \(F\), and arcs \((x_i, x_j)\) and \((x_j, x_i)\) if \(x_i = x_j\) is a constraint of \(F\). This can be done with a monotone projection. Observe that a \(\Gamma\)-formula \(F\) is unsatisfiable if, and only if, there exists a constraint of the form \(x_{i_1} \lor \cdots \lor x_{i_k}\) in \(F\), such that there exists a path from some \(x_{i_j}\) to a constraint \(\neg y\) in \(F\). This can be checked in monotone \(\text{NL}\) by Theorem B.1.

The case \(S_{10} \subseteq \text{Pol}(S)\) is analogous. \(\square\)

\section{Background on Post’s Lattice and Clones}

In this section, we include the definitions of the various clones that are used in the paper, as well as a figure of Post’s lattice, which can be helpful when checking the proofs of Section 5.

Let \(\rightarrow: (x, y) \mapsto (\neg x \lor y)\). Let also \(\leftrightarrow: (x, y) \mapsto \neg(x \lor y)\) and \(\text{id}: x \mapsto x\). Let \(f: \{0, 1\}^k \rightarrow \{0, 1\}\) be a Boolean function. We say that \(f\) is linear if there exists \(c \in \{0, 1\}^k\) and \(b \in \{0, 1\}\) such that \(f(x) = c \cdot x + b \mod 2\). We say that \(f\) is self-dual if \(f = \text{dual}(f)\). Let \(a \in \{0, 1\}\). We say that \(f\) is \(a\)-reproducing if \(f(a, \ldots, a) = a\). We say that a set \(T \subseteq \{0, 1\}^k\) is \(a\)-separating if there exists \(i \in [k]\) such that \(x_i = a\) for all \(x \in T\). We say that \(f\) is \(a\)-separating if \(f^{-1}(a)\) is \(a\)-separating. We say that \(f\) is \(a\)-separating of degree \(k\) if every \(T \subseteq f^{-1}(a)\) such that \(|T| = k\) is \(a\)-separating. The basis of a clone \(B\) is a set of Boolean functions \(F\) such that \(B = [F]\).
<table>
<thead>
<tr>
<th>Name (S_i^n)</th>
<th>Definition</th>
<th>Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>BF</td>
<td>All Boolean functions</td>
<td>(\lor, \land, \neg)</td>
</tr>
<tr>
<td>(R_0)</td>
<td>({f \in BF : f \text{ is 0-reproducing}})</td>
<td>(\land, \lor)</td>
</tr>
<tr>
<td>(R_1)</td>
<td>({f \in BF : f \text{ is 1-reproducing}})</td>
<td>(\lor, \land)</td>
</tr>
<tr>
<td>(R_2)</td>
<td>(R_1 \cap R_0)</td>
<td>(\lor, x \land (y \leftrightarrow z))</td>
</tr>
<tr>
<td>M</td>
<td>({f \in BF : f \text{ is monotonic}})</td>
<td>(\lor, \land, 0, 1)</td>
</tr>
<tr>
<td>(M_1)</td>
<td>(M \cap R_1)</td>
<td>(\lor, \land, 1)</td>
</tr>
<tr>
<td>(M_0)</td>
<td>(M \cap R_0)</td>
<td>(\lor, \land, 0)</td>
</tr>
<tr>
<td>(M_2)</td>
<td>(M \cap R_2)</td>
<td>(\lor, \land)</td>
</tr>
<tr>
<td>(S_0^n)</td>
<td>({f \in BF : f \text{ is 0-separating of degree} n})</td>
<td>(\to, \text{dual}(h_n))</td>
</tr>
<tr>
<td>(S_0)</td>
<td>({f \in BF : f \text{ is 0-separating}})</td>
<td>(\to)</td>
</tr>
<tr>
<td>(S_1^n)</td>
<td>({f \in BF : f \text{ is 1-separating of degree} n})</td>
<td>(x \land \bar{y}, h_n)</td>
</tr>
<tr>
<td>(S_1)</td>
<td>({f \in BF : f \text{ is 1-separating}})</td>
<td>(x \land \bar{y})</td>
</tr>
<tr>
<td>(S_0^n)</td>
<td>({f \in BF : f \text{ is} n\text{-ary OR function}})</td>
<td>(x \lor (y \land \bar{z}), \text{dual}(h_n))</td>
</tr>
<tr>
<td>(S_0)</td>
<td>({f \in BF : f \text{ is} n\text{-ary AND function}})</td>
<td>(\lor, \land)</td>
</tr>
<tr>
<td>(S_0^n)</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary OR function}})</td>
<td>(\lor, 0, 1)</td>
</tr>
<tr>
<td>(S_0)</td>
<td>({f \in BF : f \text{ is} n\text{-ary AND function}})</td>
<td>(\land, 0, 1)</td>
</tr>
<tr>
<td>(S_1^n)</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land, 0)</td>
</tr>
<tr>
<td>(S_1)</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land, 1)</td>
</tr>
<tr>
<td>(S_{01})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{11})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{00})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{10})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{02})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{01})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{11})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{10})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{02})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{01})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
<tr>
<td>(S_{11})</td>
<td>({f \in BF : f \text{ is constant or an} n\text{-ary AND function}})</td>
<td>(\land)</td>
</tr>
</tbody>
</table>

**Figure 4:** Table of all closed classes of Boolean functions, and their bases. Here, \(h_n\) denotes the function \(h_n(x_1, \ldots, x_{n+1}) = \bigvee_{j=1}^{n+1} \bigwedge_{i=1, i \neq j}^{n+1} x_j\). See Definition A.2 for the definition of \(\text{dual}(\cdot)\). The same table appears in [ABI+09, Table 1].