Constant-Depth Circuits vs. Monotone Circuits

Bruno P. Cavalar
University of Warwick, Coventry, UK

Igor C. Oliveira
University of Warwick, Coventry, UK

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 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 [49] and Ajtai and Gurevich [1]. In addition, our separation holds for a monotone graph property, which was unknown even in the context of AC^0 versus 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 [32].

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 AC^0 circuit of size s and depth d can be computed by a monotone circuit of size $2^{n-n/O(\log s)^{d-1}}$. We show that the existence of significantly faster monotone simulations would lead to breakthrough circuit lower bounds. In particular, if every monotone function in AC^0 admits a polynomial size monotone circuit, then NC^2 is not contained in 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 Göös, Kamath, Robere and Sokolov [31] via lifting techniques. Adapting results of Schaefer [67] and Allender, Bauland, Immerman, Schnoor and Vollmer [4], 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.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases circuit complexity, monotone circuit complexity, bounded-depth circuis, constraint-satisfaction problems

Digital Object Identifier 10.4230/LIPIcs.CCC.2023.29

Related Version *Full Version*: `https://eccc.weizmann.ac.il/report/2023/069/`

Funding This work received support from the Royal Society University Research Fellowship URF\R1\191059, the EPSRC New Horizons Grant EP/V048201/1, and the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick.

Acknowledgements We thank Arkadev Chattopadhyay for several conversations about the AC^0 versus $\mathsf{mSIZE[poly]}$ problem and related questions. We are also grateful to Denis Kuperberg for explaining to us the results from [46, 47]. 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.

1 Introduction

A Boolean function $f \colon \{0,1\}^n \to \{0,1\}$ 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[1] that compute them, have been extensively investigated for decades due to their relevance in circuit complexity [58], cryptography [10], learning theory [14], proof complexity [45, 54], property testing [28], pseudorandomness [22], optimisation [30], hazard-free computations [37], and meta-complexity [36], 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 [26] 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 [59] for Matching and the constant-depth circuit lower bound of Rossman [64] 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, [59, 8, 7, 69] showed the existence of monotone functions that can be computed by circuits of polynomial size but require monotone circuits of size $2^{n^{\Omega(1)}}$. In other words, the use of negations can significantly speedup the computation of monotone functions. More recently, Göös, Kamath, Robere and Sokolov [31] considerably strengthened this separation by showing that some monotone functions in 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 [57].) 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 [49] and (independently) Ajtai and Gurevich [1] exhibited monotone functions in AC^0 (i.e., constant-depth poly-size AND/OR/NOT circuits) that require monotone AC^0 circuits (composed of only AND/OR gates) of super-polynomial size.[2] This result has been extended to an exponential separation in [24], which shows the existence of a monotone function in AC^0 that requires monotone depth-d circuits of size $2^{\widetilde{\Omega}(n^{1/d})}$ even if MAJ (majority) gates are allowed in addition to AND/OR gates.[3]

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., [50,

[1] Recall that in a monotone Boolean circuit the gate set is limited to $\{\mathsf{AND}, \mathsf{OR}\}$ and input gates are labelled by elements from $\{x_1, \ldots, x_n, 0, 1\}$.

[2] We refer to [13] for an alternate exposition of this result.

[3] Separations between monotone and non-monotone devices have also been extensively investigated in other settings. This includes average-case complexity [12], different computational models, such as span programs [9, 62] and algebraic complexity (see [21] and references therein), and separations in first-order logic [68, 46, 47]. We restrict our attention to worst-case separations for Boolean circuits in this paper.

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 [29]. 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 [55] 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 [63, 66] showed that any homomorphism-preserving graph property computed by AC^0 circuits is also computed by monotone AC^0 circuits.[4] Under no circuit depth restriction, Berkowitz [11] proved that the monotone and non-monotone circuit size complexities of every slice function are polynomially related.[5]

Despite much progress and sustained efforts, these two classes of results leave open tantalising problems about the power of cancellations in computation.[6] 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 [32] 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 $\mathsf{AC}^0 \cap \mathsf{Mono} \nsubseteq \mathsf{mSIZE}[\mathsf{poly}]$. 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 $d\colon \mathbb{N} \to \mathbb{N}$, let $\mathsf{mDEPTH}[d]$ denote the class of Boolean functions computed by monotone fan-in two AND/OR Boolean circuits of depth $O(d(n))$. Similarly, we use $\mathsf{mSIZE}[s]$ 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 $\mathsf{m}\mathcal{C}$ denote its natural monotone analogue. Finally, for a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we use $\mathsf{mSIZE}(f)$ and $\mathsf{mDEPTH}(f)$ to denote its monotone circuit size and depth complexities, respectively. We refer to Jukna [42] for standard background on circuit complexity theory.

[4] A function $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ 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.

[5] A function $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ 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.

[6] Any non-monotone circuit can be written as an XOR (parity) of distinct monotone sub-circuits (see, e.g., [33, 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 [71].

1.1.1 Constant-depth circuits vs. monotone circuits

Recall that the Okol'nishnikova-Ajtai-Gurevich [49, 1] theorem states that $\mathsf{AC}^0 \cap \mathsf{Mono} \not\subseteq \mathsf{mAC}^0$. 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** (Polynomial-size constant-depth vs. larger monotone depth). *For every $k \geq 1$, we have $\mathsf{AC}^0 \cap \mathsf{Mono} \not\subseteq \mathsf{mDEPTH}[(\log n)^k]$. Moreover, this separation holds for a monotone graph property.*

In a more constrained setting, Kuperberg [46, 47] 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 AC^0 versus mAC^0.

Let HomPreserving denote the class of all homomorphism-preserving graph properties, and recall that Rossman [63, 66] established that $\mathsf{AC}^0 \cap \mathsf{HomPreserving} \subseteq \mathsf{mAC}^0$. Theorem 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 [49, 1, 13, 24] and their counterparts in first-order logic [68, 46, 47]. In particular, it allows us to break the $O(\log n)$ monotone depth barrier present in previous separations with an AC^0 upper bound, which rely on lower bounds against monotone circuits of depth d and size (at most) $2^{n^{O(1/d)}}$. We defer the discussion of our techniques to Section 1.2.

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

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

Theorem 1 and Theorem 2 are incomparable: while the monotone lower bound is stronger in the latter, its constant-depth upper bound requires parity gates. Theorem 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 [32].

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 and Theorem 2 provide more evidence for the existence of monotone functions in 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 $\mathsf{AC}^0_d[s]$ denote the class of Boolean functions computed by 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 $\mathsf{AC}^0_d[s] \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{n(1-1/O(\log s)^{d-1})}]$ (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 $d \leq 2$ [55].) 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 3** (New circuit lower bounds from monotone simulations). *There exists $\varepsilon > 0$ such that the following holds.*

1. *If* $\mathsf{AC}^0_3 \cap \mathsf{Mono} \subseteq \mathsf{mNC}^1$, *then* $\mathsf{NP} \not\subseteq \mathsf{AC}^0_3[2^{o(n)}]$.
2. *If* $\mathsf{AC}^0_4 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, *then* $\mathsf{NP} \not\subseteq \mathsf{AC}^0_4[2^{o(\sqrt{n}/\log n)}]$.
3. *If* $\mathsf{AC}^0 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, *then* $\mathsf{NC}^2 \not\subseteq \mathsf{NC}^1$.
4. *If* $\mathsf{NC}^1 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{O(n^\varepsilon)}]$, *then* $\mathsf{NC}^2 \not\subseteq \mathsf{NC}^1$.
5. *If* $\mathsf{AC}^0 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, *then* $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.
6. *If* $\mathsf{NC}^1 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, *then* $\mathsf{L} \not\subseteq \mathsf{NC}^1$.

Item (3) of Theorem 3 implies in particular that, if the upper bound of Theorem 2 cannot be improved to AC^0 (i.e., the question asked by [32] has a negative answer), then $\mathsf{NC}^2 \not\subseteq \mathsf{NC}^1$. It also improves a result from [23] showing the weaker conclusion $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ 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 [63, 66] achieves this for graph properties that are preserved under homomorphisms. Items (5) and (6) of Theorem 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 [31] showed the existence of a monotone function f^{GKRS} in NC^2 that is not in $\mathsf{mSIZE}[2^{n^{\Omega(1)}}]$. As opposed to classical results [59, 8, 7, 69] 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 $\mathsf{mSIZE}[\mathsf{poly}]$.

As remarked in [31, 26], 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 $\mathsf{CSP\text{-}SAT}_S$ which capture the framework and lower bound of [31].

Encoding CSPs as monotone Boolean functions. Let $R \subseteq \{0,1\}^k$ be a relation. We call k the *arity* of R. Let $V = (i_1, \dots, i_k) \in [n]^k$, and let $f_{R,V} : \{0,1\}^n \to \{0,1\}$ be the function that accepts a string $x \in \{0,1\}^n$ if $(x_{i_1}, \dots, x_{i_k}) \in R$. 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 $S = \{R_1, \dots, R_k\}$ be a finite set of Boolean relations. Let ℓ_i be the arity of the relation R_i. Note that there are n^{ℓ_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 $\mathsf{CSP\text{-}SAT}^n_S : \{0,1\}^N \to \{0,1\}$ 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 $\{\mathsf{CSP\text{-}SAT}^n_S\}_{n \geq 1}$ as a sequence of functions, we simply write $\mathsf{CSP\text{-}SAT}_S$.

The function f^{GKRS} from [31] is simply $\mathsf{CSP\text{-}SAT}_S$ for $S = \{\oplus^0_3, \oplus^1_3\}$, where we write $\oplus^b_3(x_1, x_2, x_3) = 1$ if and only if $\sum_i x_i = b \pmod 2$. 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 $\mathsf{CSP\text{-}SAT}^n_S$, where $n = \mathsf{poly}(m)$.

Despite the generality of the technique from [31] and the vast number of possibilities for S, we prove that a direct application of their approach cannot establish Theorem 1 and Theorem 2. This is formalised as follows. (We refer to Section 5 for much stronger forms of the result.)

▶ **Theorem 4** (Limits of the direct approach via lifting and CSPs). *Let S be a finite set of Boolean relations. The following holds.*
1. *If* $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mSIZE}[\mathsf{poly}]$ *then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\oplus\mathsf{L}$*-hard under* $\leq^{\mathsf{AC}^0}_m$ *reductions.*
2. *If* $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mNC}^1$ *then* $\mathsf{CSP\text{-}SAT}_S$ *is* L*-hard under* $\leq^{\mathsf{AC}^0}_m$ *reductions.*

In particular, since there are functions (e.g., Majority) computable in logarithmic space that are not in $\mathsf{AC}^0[\oplus]$, Theorem 4 (Part 2) implies that any $\mathsf{CSP\text{-}SAT}_S$ function that is hard for poly-size monotone formulas (mNC^1) must lie outside $\mathsf{AC}^0[\oplus]$. Observe that this can also be interpreted as a *monotone simulation*: for any finite set S of Boolean relations, if $\mathsf{CSP\text{-}SAT}_S \in \mathsf{AC}^0[\oplus]$ then $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNC}^1$.[7]

Theorem 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 $\mathsf{Pol}(S)$ of *polymorphisms* of S, a standard concept in the investigation of CSPs.[8] We present next a simplified 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 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* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$ *there is* $\varepsilon > 0$ *such that* $\mathsf{mSIZE}(\mathsf{CSP\text{-}SAT}_S) = 2^{\Omega(n^\varepsilon)}$. *Otherwise,* $\mathsf{mSIZE}(\mathsf{CSP\text{-}SAT}_S) = n^{O(1)}$.
2. Monotone Depth Dichotomy: *If* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$ *or* $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$ *or* $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$*, there is* $\varepsilon > 0$ *such that* $\mathsf{mDEPTH}(\mathsf{CSP\text{-}SAT}_S) = \Omega(n^\varepsilon)$. *Otherwise,* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNC}^2$.

[7] Jumping ahead, our proof of Theorem 2 still relies in a crucial way on the monotone lower bound obtained by [31]. However, our argument requires an extra ingredient and does not follow from a direct application of their template. We provide more details about it in Section 1.2 below. Interestingly, the proof of Theorem 1 was discovered by trying to avoid the "barrier" posed by Theorem 4.

[8] Roughly speaking, $\mathsf{Pol}(S)$ captures the amount of symmetry in S, and a larger set $\mathsf{Pol}(S)$ implies that solving $\mathsf{CSP\text{-}SAT}_S$ 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.

We note that previous papers of Schaefer [67] and Allender, Bauland, Immerman, Schnoor and Vollmer [4] provided a *conditional* classification of the complexity of such CSPs. Theorem 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, which appears last. We discuss this result after explaining the proof of Theorem 2 and the classification of the monotone complexity of CSPs (Theorem 4 and Theorem 5), as this sheds light into how the proof of Theorem 1 was discovered and into the nature of the argument.

A monotone circuit size lower bound for a function in $\mathsf{AC}^0[\oplus]$. We first give an overview of the proof of Theorem 2.

The lower bound of [31]. We begin by providing more details about the aforementioned monotone circuit lower bound of [31], since their result is a key ingredient in our separation (see [26] for a more detailed overview). Recall that their function f^{GKRS} corresponds to $\mathsf{CSP\text{-}SAT}_S$ for $S = \{\oplus_3^0, \oplus_3^1\}$. Following their notation, this is simply the Boolean function $\mathsf{3\text{-}XOR\text{-}SAT}_n \colon \{0,1\}^{2n^3} \to \{0,1\}$ 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, [31] employed a lifting technique from communication complexity to show the existence of a constant $\varepsilon > 0$ such that $\mathsf{mSIZE}(\mathsf{3\text{-}XOR\text{-}SAT}_n) = 2^{n^\varepsilon}$. (We show in Appendix A that a weaker super-polynomial monotone circuit size lower bound for $\mathsf{3\text{-}XOR\text{-}SAT}_n$ can also be obtained using the approximation method and a reduction.)

Sketch of the proof of Theorem 2. Since $\mathsf{3\text{-}XOR\text{-}SAT}_n \in \mathsf{NC}^2$ (see, e.g, [31]), their result implies that $\mathsf{NC}^2 \cap \mathsf{Mono} \nsubseteq \mathsf{mSIZE}[2^{n^{\Omega(1)}}]$. On the other hand, we are after a separation between *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 [31] 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 $\mathsf{CSP\text{-}SAT}_S$ while retaining super-polynomial monotone hardness. However, Theorem 4 shows that this is impossible, as explained above.

A second possibility is to combine the *exponential* 2^{n^ε} monotone circuit size lower bound for $\mathsf{3\text{-}XOR\text{-}SAT}_n$ and a padding argument, since we only need *super-polynomial* hardness. Indeed, this argument can be used to define a monotone function $g \colon \{0,1\}^n \to \{0,1\}$ that is computed by polynomial-size fan-in two circuits of depth $\mathsf{poly}(\log\log n)$ 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.

[9] We remark that only recently has Schaefer's classification been extended to the non-Boolean case [72, 15]. Though the refined classification of [4] is conjectured to hold analogously in the case of non-Boolean CSPs [48], this is still open (see the discussion in [16, Section 7]).

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 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 [31] and other papers, is that $\mathsf{3\text{-}XOR\text{-}SAT}_n$ 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 [44], which correspond to the non-uniform version of $\oplus\mathsf{L}$, 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 $\mathsf{AC}^0[\oplus]$ 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 $\oplus\mathsf{L}$ using $\mathsf{AC}^0[\oplus]$ circuits (this folklore result appears implicit in [6] and [51]). Putting everything together, it follows that for a large enough but constant depth, $\mathsf{3\text{-}XOR\text{-}SAT}_n$ can be computed by $\mathsf{AC}^0[\oplus]$ circuits of size $2^{n^{\varepsilon/2}}$. Since this function is hard against monotone circuits of size $2^{n^{\varepsilon}}$, a padding argument can now be used to establish a separation between $\mathsf{AC}^0[\oplus]$ and $\mathsf{mSIZE}[\mathsf{poly}]$. (A careful choice of parameters provides the slightly stronger statement in Theorem 2.)

Non-trivial monotone simulations and their consequences. In order to conclude that significantly stronger monotone simulations imply new complexity separations (Theorem 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 $\mathsf{NC}^2 \subseteq \mathsf{NC}^1$, then $\mathsf{3\text{-}XOR\text{-}SAT} \in \mathsf{NC}^1$, and we can conclude by standard depth-reduction for NC^1 and padding, together with the exponential lower bound for $\mathsf{3\text{-}XOR\text{-}SAT}$ due to [31], 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 [23] 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 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 structure (Lemma 12). We then argue as in the case for general functions, using known monotone lower bounds for graph properties. We note that Lemma 12 is also important in the proof of Theorem 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 4, Theorem 5, and the extensions discussed in Section 5.

A seminal work of Schaefer [67] proved that any Boolean CSP is either solvable in polynomial-time or it is NP-complete. Later, Jeavons [38] observed that the complexity of deciding if a given set of constraint applications of S is satisfiable depends exclusively on the set $\mathsf{Pol}(S)$ 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 $\mathsf{Pol}(S) \subseteq \mathsf{Pol}(S')$, 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 $\mathsf{CSP\text{-}SAT}_S$, in the way we described above. We reexamine Schaefer's and Jeavons's proofs and establish that the reduction from $\mathsf{CSP\text{-}SAT}_{S'}$ to $\mathsf{CSP\text{-}SAT}_S$ can also be done with efficient monotone circuits. Making use of and adapting parts of the refined results and analysis of [4], which builds on the earlier dichotomy result of [67] 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 [53, 18, 19], we are able to show that $\mathsf{CSP\text{-}SAT}_S$ is hard for monotone circuits only when $\mathsf{CSP\text{-}SAT}_S$ is $\oplus\mathsf{L}$-complete, as in Theorem 4 Part 1.

A monotone circuit depth lower bound for a function in AC^0. Next, we combine insights obtained from the monotone lower bound of [31], our proof of Theorem 2 via a guess-and-verify depth reduction and padding, and the statement of Theorem 4 (limits of the direct approach via CSPs) to get the separation in Theorem 1. As alluded to above, our approach differs from those of [49, 1, 13, 24] and related results in the context of first-order logic [68, 46, 47].

Recall that the [31] framework lifts a Resolution width lower bound for an unsatisfiable S-formula F into a corresponding monotone circuit size lower bound for $\mathsf{CSP\text{-}SAT}_S$. On the other hand, Theorem 4 rules out separating constant-depth circuits from monotone circuits of polynomial size via $\mathsf{CSP\text{-}SAT}_S$ functions. In particular, we cannot directly apply the chain of reductions from [31] 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 $\mathsf{AC}^0[\oplus]$ upper from Theorem 2 to the desired AC^0 upper bound in Theorem 1.

In [31] 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 $\mathsf{CSP\text{-}SAT}_S$ 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 $\oplus\mathsf{L}$, which is the main bottleneck in the proof of Theorem 2 yielding $\mathsf{AC}^0[\oplus]$ circuits instead of AC^0. At the same time, considering this structural property immediately takes us outside the domain of Theorem 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 $\mathsf{OddFactor}_n : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ 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 $\mathsf{OddFactor}_n$ 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 $\mathsf{CSP\text{-}SAT}_S$ function obtained from an appropriate Tseitin formula via the framework of [31]) admits a $\oplus\mathsf{L}$ upper bound, we observe that $\mathsf{OddFactor}_n$ 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 [60]. (We note that related ideas appear in an unpublished note of Johannsen [40].) This is the first application of Reingold's algorithm to this kind of separation.

At the same time, $\mathsf{OddFactor}_n$ retains at least part of the monotone hardness of 3-XOR-SAT. Using a different reduction from a communication complexity lower bound, [9] proved that the monotone circuit depth of $\mathsf{OddFactor}_n$ is $n^{\Omega(1)}$. Altogether, we obtain a monotone Boolean function (indeed a graph property) that lies in L but is not in $\mathsf{mDEPTH}[n^{o(1)}]$. Applying a guess-and-verify depth reduction for L and using (graph) padding (analogously to the proof sketch of Theorem 2), we get a monotone graph property in AC^0 that is not in $\mathsf{mDEPTH}[\log^k n]$. This completes the sketch of the proof of Theorem 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 [55] 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 19 (see Section 4.1) implies that every monotone function in depth-3 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 [24] rules out an efficient *constant-depth* monotone simulation, it is still possible (and consistent with Theorem 1) that $\mathsf{AC}^0_3 \cap \mathsf{Mono} \subseteq \mathsf{mNC}^1$. 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 [32]. Using our approach, it would be sufficient to show that $\mathsf{OddFactor}_n$ 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 $\mathsf{Matching}_n$, a longstanding open problem in monotone complexity (see [42, Section 9.11]).[11] Indeed, it's possible to reduce OddFactor to Matching using monotone AC^0 circuits (see [3, Lemma 6.18]).

Incidentally, the algebraic complexity variant of the AC^0 vs. $\mathsf{mSIZE}[\mathsf{poly}]$ problem has been recently settled in a strong way through a new separation result obtained by Chattopadhyay, Datta, and Mukhopadhyay [21]. Could some of their techniques be useful to attack the more elusive Boolean case?

[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.

[11] 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.

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.

2 Preliminaries

2.1 Notation

Boolean functions. We denote by Mono the set of all monotone Boolean functions. We define $\mathsf{poly} = \{n \mapsto n^C : C \in \mathbb{N}\}$. A Boolean function $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ 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 $\mathsf{HomPreserving}$ the set of all graph properties which are preserved under homomorphisms. Note that $\mathsf{HomPreserving} \subseteq \mathsf{Mono}$.

Boolean circuits. We denote by $\mathsf{AC}^0_d[s]$ the family of Boolean functions computed by size-s, depth-d Boolean circuits with unbounded fan-in $\{\wedge, \vee\}$-gates and input literals from $\{x_1, \overline{x_1}, \dots, x_n, \overline{x_n}\}$. We write $\mathsf{AC}^0[s]$ as a shorthand for $\bigcup_{d=1}^{\infty} \mathsf{AC}^0_d[s]$, and AC^0 as a shorthand of $\mathsf{AC}^0[n^{O(1)}] = \mathsf{AC}^0[\mathsf{poly}]$. We will also refer to $\mathsf{AC}^0_d[\mathsf{poly}]$ by AC^0_d. We write $\mathsf{DNF}[s]$ to denote the family of Boolean functions computed by size-s DNFs, where size is measured by number of terms. We write $\mathsf{CNF}[s]$ analogously. We write $\mathsf{SIZE}[s]$ to denote the family of Boolean functions computed by size-s circuits. We write $\mathsf{DEPTH}[d]$ to denote the family of Boolean functions computed by fan-in 2 circuits of depth d. We denote by $\mathsf{AC}^0[\oplus]$ the family of Boolean functions computed by polynomial-size AC^0 circuits with unbounded fan-in $\oplus$-gates.

We denote by L the family of Boolean functions computed by logspace machines, and by NL the family of Boolean functions computed by polynomial-time nondeterministic logspace machines. Moreover, we denote by $\oplus\mathsf{L}$ 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 $\mathsf{m}\mathcal{C}$ to denote the monotone version of $\mathcal{C}$. Given a function f, we write $\mathsf{mSIZE}(f)$ to denote the size of the smallest monotone circuit computing f and $\mathsf{mDEPTH}(f)$ to denote the smallest depth of a fan-in 2 monotone circuit computing f. Given two Boolean functions f, g, we write $f \leq_m^{\mathsf{mProj}} g$ 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 α. We let $\mathrm{supp}(\alpha) = \{i \in [n] : \alpha_i = 1\}$. We let $\mathrm{THR}_{k,n} : \{0,1\}^n \to \{0,1\}$ be the Boolean function such that $\mathrm{THR}_{k,n}(x) = 1 \iff |x|_1 \geq k$.

[12] A monotone projection is a projection without negations.

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 $d = O_{\varepsilon}(1)$.

▶ **Lemma 6** (Folklore; see, e.g., [5, Lemma 8.1]). *For all $\varepsilon > 0$, we have $\mathsf{NL} \subseteq \mathsf{AC}^0[2^{n^{\varepsilon}}]$.*

3 Constant-Depth Circuits vs. Monotone Circuits

In this section, we prove Theorems 1 and 2. For the upper bounds, we require the logspace graph connectivity algorithm due to [60] and the $\oplus\mathsf{L}$ algorithm for solving linear systems over $\mathbb{F}_2$ due to [17], as well as the depth-reduction techniques of [6, 5]. On the lower bounds side, our proofs rely on previous monotone circuit and depth lower bounds from [9, 31]. 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 $\mathsf{AC}^0[\oplus]$

In this section, we prove Theorem 2. We first recall the monotone circuit lower bound of [31] and a depth-reduction lemma implicit in [6] and [51], whose full proof we give below for completeness. We remark that similar arguments can be employed to prove Lemma 6, essentially by replacing the $\oplus$ gates by $\vee$ gates.

As explained in Section 1.2, in its strongest form the separation result from [31] can be stated as follows.

▶ **Theorem 7** ([31]). *There exists $\varepsilon > 0$ such that $\oplus\mathsf{L} \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{o(n^{\varepsilon})}]$. Moreover, this separation is witnessed by* 3-XOR-SAT.

▶ **Lemma 8** (Folklore; see, e.g., [6, 51]). *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a $\oplus\mathsf{L}$ machine. For every $\delta > 0$, there exists an $\mathsf{AC}^0[\oplus]$ circuit of size $2^{n^{\delta}}$ that computes f.*

Proof. Let M be a $\oplus\mathsf{L}$-machine computing f. Without loss of generality, we may 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.[13] 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_{start} to the accepting configuration v_{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_{start} to v_{accept} with a depth-d $\mathsf{AC}^0[\oplus]$ circuit. First, choose $m^{1/d} - 1$ configurations $v_1, \ldots, v_{m^{1/d}-1}$ (henceforth called

[13] Formally, we can define a $\oplus\mathsf{L}$-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'.

"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 $v_0 = v_{\mathsf{start}}$ and $v_{m^{1/d}} = v_{\mathsf{accept}}$. We then count the number of paths from from v_{start} to v_{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 $\#\mathsf{paths}(s, t, \ell)$ denote the number of paths between configurations s and t with distance exactly ℓ, we obtain

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

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 $\#\mathsf{paths}(v_i, v_{i+1}, m^{1-1/d})$ for each i, we obtain a depth-$2d$ $\mathsf{AC}^0[\oplus]$ circuit for calculating the number of paths from v_{start} to v_{accept} (modulo 2). Clearly, the total size of the circuit is $2^{O(m^{1/d} \cdot \log m)}$, which is smaller than 2^{n^δ} for a large enough constant d. ◀

We now restate Theorem 2 and prove it by combining Theorem 7 and Lemma 8 with a padding trick.

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

Proof. By Theorem 7, there exists $\varepsilon > 0$ and a monotone function $f \in \oplus\mathsf{L}$ 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 $N := m + n = 2^{\Theta(n^\delta)}$ bits. By Lemma 8, there exists an $\mathsf{AC}^0[\oplus]$ 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 $2^{\Omega(n^\varepsilon)} = 2^{\Omega((\log N)^{\varepsilon/\delta})} = 2^{\Omega((\log N)^k)}$. ◀

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

In this section, we prove Theorem 1. We prove moreover that the function that separates $\mathsf{AC}^0 \cap \mathsf{Mono}$ and mNC^i can be taken to be a graph property. We state our result in its full generality below.

▶ **Theorem 9.** *For every $i \geq 1$, we have* $\mathsf{AC}^0 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mDEPTH}[(\log n)^i]$. *In particular, we have* $\mathsf{AC}^0 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mNC}^i$.

First, we recall a result of [9], which proves monotone lower bounds for the following function. Let $\mathsf{OddFactor}_n : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ 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 [9] proved the following result:

▶ **Theorem 10** ([9])**.** *Any monotone formula computing* $\mathsf{OddFactor}_n$ *has size* $2^{\Omega(n)}$*, and any monotone circuit computing* $\mathsf{OddFactor}_n$ *has size* $n^{\Omega(\log n)}$.

The proof in [9] 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 DISJ_n [43], which were not available to [9]. We remark that, with a different language, a monotone circuit lower bound for OddFactor is also implicitly proved in Feder and Vardi [27, Theorem 30].

We now recall an upper bound for OddFactor, implicitly proved in an unpublished note due to Johannsen [40].

▶ **Theorem 11** ([40]). *We have* $\mathsf{OddFactor} \in \mathsf{L}$.

Proof. We first recall the following observation about the OddFactor function, which appears in different forms in the literature (see [70, Lemma 4.1] or [42, Lemma 18.16]; see also [40, 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 $P = (x_1, \dots, x_k)$ between u and v, where $x_1 = u$ and $x_k = v$. If the edge $x_i x_{i+1}$ 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 [60]. 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 AC^0 not computed by monotone circuits of depth $(\log n)^i$, we can follow the same argument of Theorem 2, using Lemma 6 instead of Lemma 8. 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 12.** *Let* $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ *be a monotone graph property on graphs of* n *vertices. The following holds.*

1. *If* $f \in \mathsf{NC}^i$ *for some* $i > 1$*, then there exists a monotone graph property* g *on graphs of* $N = 2^{(\log n)^i}$ *vertices such that* $g \in \mathsf{NC}^1$ *and* $f \leq_m^{\mathsf{mProj}} g$.
2. *If* $f \in \mathsf{NL}$*, 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* AC^0 *circuits of size* $N^{2+o(1)}$ *and* $f \leq_m^{\mathsf{mProj}} g$.
3. *If* $f \in \oplus\mathsf{L}$*, 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* $\mathsf{AC}^0[\oplus]$ *circuits of size* $N^{2+o(1)}$ *and* $f \leq_m^{\mathsf{mProj}} g$.

We are now ready to prove Theorem 9.

Proof of Theorem 9. Fix $n \in \mathbb{N}$ and take an $\varepsilon < 1/i$. Observing that $\mathsf{L} \subseteq \mathsf{NL}$, from Theorem 11 and item (2) of Lemma 12 we conclude that there exists a monotone graph property f on $N = 2^{n^{\varepsilon}}$ vertices such that $f \in \mathsf{AC}^0$ and $\mathsf{OddFactor}_n \leq_m^{\mathsf{mProj}} f$. By Theorem 10, any monotone circuit computing f has depth $\Omega(n) = \Omega((\log N)^{1/\varepsilon}) \gg (\log N)^i$. ◀

Raz and Wigderson [57] observed that there exists a monotone function $f \in \mathsf{NC}^1 \setminus \mathsf{mNC}$. Using Lemma 12, we observe moreover that it's possible to obtain this separation with a monotone graph property.

▶ **Proposition 13.** *We have* $\mathsf{NC}^1 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mNC}$.

Proof. Observing that $\mathsf{L} \subseteq \mathsf{NC}^2$, we conclude from Theorem 11 and item (1) of Lemma 12 that there exists a monotone graph property f on $N = 2^{(\log n)^2}$ vertices such that $f \in \mathsf{NC}^1$ and $\mathsf{OddFactor}_n \leq_m^{\mathsf{mProj}} f$. By Theorem 10, any monotone circuit computing f has depth $\Omega(n) = \Omega(2^{\sqrt{\log N}})$, which implies $f \notin \mathsf{mNC}$. ◀

3.3 Efficient monotone padding for graph properties

We will now prove Lemma 12. 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 14** ([35]). *Let $d > 0$ be a constant. The function $\mathrm{THR}_{(\log n)^d, n}$ can be computed by an AC^0 circuit of size $n^{o(1)}$ and depth $d + O(1)$.*

▶ **Theorem 15** ([2]). *For every $k \in [n]$, the function $\mathrm{THR}_{k,n}$ can be computed by a circuit of depth $O(\log n)$ and size $n^{O(1)}$.*

▶ **Lemma 16.** *There exists a circuit C_n^k 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 $k = \mathrm{polylog}(n)$, 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

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

We first consider the case $k = \mathrm{polylog}(n)$. In this case, the first two conditions can be checked with circuits of size $n^{o(1)}$ using Theorem 14. 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 NC^1 circuit by Theorem 15. 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 12.

Proof of Lemma 12. We first prove (1). Fix $n \in \mathbb{N}$ and let $N = 2^{(\log n)^i}$. For a graph G on N vertices such that $|E(G)| \leq \binom{n}{2}$, let G_{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_{clean}, *or* (2) G_{clean} has exactly n vertices. Let $g : \{0,1\}^{\binom{N}{2}} \to \{0,1\}$ be the monotone graph property defined as follows:

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

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_{clean} has exactly n vertices and $f(G_{\mathsf{clean}}) = 1$, or
2. G has more than $\binom{n}{2}$ edges, or
3. G_{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_{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 by 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 $G'_{\mathsf{clean}} = G$ (up to isomorphism) and $g(G') = f(G_{\mathsf{clean}}) = f(G)$.

We now show how to compute g in 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_{clean} and output $f(G_{\mathsf{clean}})$.

Note that checking if $|E(G)| > \binom{n}{2}$ can be done in NC^1 by Theorem 15. Moreover, for all $i \in [N]$, we have $\alpha_i = \bigvee_{j \in [N]} x_{ij}$, and therefore α_i can be computed by a circuit of depth $O(\log N)$ and $O(N)$ gates. In total, the vector α can be computed with $O(N^2)$ gates and $O(\log N)$ depth. Finally, we can check if $|\alpha|_1 > n$ in NC^1 with a threshold circuit.

For the final step, we compute G_{clean}. If $|\alpha|_1 = n$, note that $G_{\mathsf{clean}} = G[\mathrm{supp}(\alpha)]$. When $|\alpha|_1 < n$, then G_{clean} is $G[\mathrm{supp}(\alpha)]$ padded with isolated vertices. We can therefore compute G_{clean} with the circuit C_N^n of Lemma 16. Moreover, since $f \in \mathsf{NC}^i$, we have that f can be computed by a circuit of size $n^{O(1)} = N^{o(1)}$ and depth $O((\log n)^i) = O(\log N)$. Therefore, computing $f(G_{\mathsf{clean}})$ can be done in NC^1. Overall, we get that $g \in \mathsf{NC}^1$.

In order to prove (2), it suffices to modify the proof above. The modification can be briefly described as follows. We let $N = 2^{n^{\varepsilon}}$. Every time Lemma 16 is applied, we use the AC^0 circuit instead of the NC^1 circuit, since $n = \mathrm{polylog}(N)$. This ammounts to $N^{2+o(1)}$ many gates with unbounded fan-in. Moreover, since by assumption $f \in \mathsf{NL}$, applying Lemma 6 we obtain an AC^0 circuit for f of size $2^{n^{\varepsilon/2}} = N^{o(1)}$, so we can compute $f(G_{\mathsf{clean}})$ 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 6 by an application of Lemma 8. ◀

4 Non-Trivial Monotone Simulations and Their Consequences

In contrast to Section 3, in this section we observe that a non-trivial simulation of AC^0 circuits by monotone circuits is possible. This follows from a refined version of the switching lemma proved by Rossman [65]. As a proof of concept, we use this simulation result to reprove a well-known 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 [63], 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 6 and 12).

4.1 A non-trivial simulation for bounded-depth circuits

The earliest monotone simulation result was proved for DNFs by Quine [55].

▶ **Theorem 17** (Quine [55]). *For all $s : \mathbb{N} \to \mathbb{N}$, we have $\mathsf{DNF}[s] \cap \mathsf{Mono} \subseteq \mathsf{mDNF}[s]$.*

Proof. If a given DNF computes a monotone Boolean function, simply removing the negative literals continues to compute the same function. ◀

Let $\mathsf{DT}_{\mathsf{size}}(f)$ denote the size of a smallest decision-tree computing f. We will need a result obtained by Rossman [65].

▶ **Theorem 18** ([65]). *If $f : \{0,1\}^n \to \{0,1\}$ is computable by an AC^0 circuit of depth d and size s, then $\mathsf{DT}_{\mathsf{size}}(f) = 2^{(1-1/O(\log s)^{d-1})n}$.*

▶ **Theorem 19.** *Let $s : \mathbb{N} \to \mathbb{N}$ and $d \geq 1$. We have $\mathsf{AC}^0_d[s] \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[t]$, 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 AC^0 circuit of depth d and size s. By Theorem 18, 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 by Theorem 17. This can be converted into a monotone circuit of size $n \cdot 2^{(1-1/O(\log s)^{d-1})n}$. ◀

We observe that it is possible to immediately deduce an 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 [34] and the proof of the main technical tool (Theorem 18) behind our simulation result is similar to the one used by [34], the argument below illustrates how a monotone simulation can lead to non-monotone circuit lower bounds.

▶ **Corollary 20.** *Any depth-d 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 19, it follows that the size s of a depth-d 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 21.** *For all constants $\varepsilon > 0$ and $C \geq 1$, if $\mathsf{AC}^0 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{O((\log n)^C)}]$, then $\mathsf{NL} \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{o(n^\varepsilon)}]$.*

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

Let $\delta = \varepsilon/(2C)$ and let $m = 2^{n^\delta}$. Let $g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be the Boolean function defined as $g(x,y) = f(x)$. Note that g is a function on $N := m + n = 2^{\Theta(n^\delta)}$ bits. By Lemma 6, there exists an AC^0 circuit computing f of size $2^{n^\delta} = N^{O(1)}$. Moreover, any monotone circuit computing g has size $2^{\Omega(n^\varepsilon)} = 2^{\Omega((\log N)^{\varepsilon/\delta})} = 2^{\Omega((\log N)^{2C})}$. ◀

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

▶ **Theorem 22** ([52]). $\mathsf{NP} \cap \mathsf{Mono} \not\subseteq \mathsf{mDEPTH}[o(n)]$.

▶ **Theorem 23** ([20]). $\mathsf{NP} \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{o(\sqrt{n}/\log n)}]$.

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 $2^{\Omega(n^{1/3})}$ [34]. Moreover, no strict separation is known in the following sequence of inclusions of complexity classes: $\mathsf{ACC} \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \oplus\mathsf{L} \subseteq \mathsf{NC}^2$. 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.)[14]

▶ **Theorem 24.** *Let $\mathcal{C}$ be a class of circuits. There exists $\varepsilon > 0$ such that the following holds:*
1. *If $\mathsf{AC}^0_3 \cap \mathsf{Mono} \subseteq \mathsf{mNC}^1$, then $\mathsf{NP} \not\subseteq \mathsf{AC}^0_3[2^{o(n)}]$.*
2. *If $\mathsf{AC}^0_4 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, then $\mathsf{NP} \not\subseteq \mathsf{AC}^0_4[2^{o(\sqrt{n}/\log n)}]$.*
3. *If $\mathcal{C} \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{O(n^\varepsilon)}]$, then $\mathsf{NC}^2 \not\subseteq \mathcal{C}$.*
4. *If $\mathsf{AC}^0 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, then $\mathsf{NC}^2 \not\subseteq \mathsf{NC}^1$.*

Proof. We will prove each item separately.

Proof of 1. Let us assume that $\mathsf{AC}^0_3 \cap \mathsf{Mono} \subseteq \mathsf{mNC}^1$. Let f be the function of Theorem 22. For a contradiction, suppose that $f \in \mathsf{AC}^0_3[2^{o(n)}]$. Let $\alpha : \mathbb{N} \to \mathbb{N}$ be such that $\alpha(n) \to_n \infty$ and f has a depth-3 AC^0 circuit of size $2^{n/\alpha}$. Let $m = 2^{n/(10\cdot\alpha)}$ and let $g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be the function $g(x,y) = f(x)$. Let $N = n + m = (1+o(1))2^{n/(10\cdot\alpha)}$. 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 $f(x) = g(x, 1^m)$, we obtain a monotone formula of size $N^{O(1)} = 2^{o(n)}$ for computing f, which contradicts the lower bound of Theorem 22.

[14] In 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 2. Similar to the proof of item (1), but using Theorem 23 instead.

Proof of 3. Suppose that $\mathsf{NC}^2 \subseteq \mathcal{C}$. By Theorem 7, there exists a monotone function $f \in \mathsf{NC}^2$ on n bits and a number $\varepsilon > 0$ such that $f \notin \mathsf{mSIZE}[2^{o(n^\varepsilon)}]$. Therefore, for any $\delta > 0$ such that $\delta < \varepsilon$, we have $f \notin \mathsf{mSIZE}[2^{O(n^\delta)}]$. Since, by assumption, we have $f \in \mathsf{NC}^2 \subseteq \mathcal{C}$, we obtain $\mathcal{C} \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{O(n^\delta)}]$.

Proof of 4. If $\mathsf{NC}^2 \subseteq \mathsf{NC}^1$, then, by item (3), we get $\mathsf{NC}^1 \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{o(n^\varepsilon)}]$. From Lemma 21, we obtain $\mathsf{AC}^0 \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[\mathsf{poly}]$. ◀

As a motivation to the ensuing discussion, we recall a result of Rossman [63], who showed that any homomorphism-preserving graph property computed by AC^0 circuits is also computed by monotone AC^0 circuits.

▶ **Theorem 25** ([63]). $\mathsf{AC}^0 \cap \mathsf{HomPreserving} \subseteq \mathsf{mDNF}[\mathsf{poly}]$.

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 26** ([7]). *There exists $\varepsilon > 0$ such that $\mathsf{NP} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mSIZE}[2^{o(n^\varepsilon)}]$.*

▶ **Theorem 27.** *Let $\mathcal{C}$ be a class of circuits. The following holds:*

1. *If $\mathcal{C} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, then $\mathsf{L} \not\subseteq \mathcal{C}$.*
2. *If $\mathcal{C} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mDEPTH}[o(\sqrt{n})]$, where n denotes the number of input bits, then $\mathsf{L} \not\subseteq \mathcal{C}$.*
3. *If $\mathsf{AC}^0 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.*

Proof. We will prove each item separately.

Proof of 1. Suppose that $\mathsf{L} \subseteq \mathcal{C}$. By Theorem 10, the monotone graph property $\mathsf{OddFactor}$ satisfies $\mathsf{OddFactor} \notin \mathsf{mSIZE}[\mathsf{poly}]$. Moreover, we have the upper bound $\mathsf{OddFactor} \in \mathsf{L}$ by Theorem 11. Since, by assumption, we have $\mathsf{OddFactor} \in \mathsf{L} \subseteq \mathcal{C}$, we obtain $\mathcal{C} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mSIZE}[\mathsf{poly}]$.

Proof of 2. Suppose that $\mathsf{L} \subseteq \mathcal{C}$. By Theorems 10 and 11, there exists a monotone graph property $f \in \mathsf{L}$ such that $f \notin \mathsf{mDEPTH}[o(\sqrt{n})]$. Since, by assumption, we have $f \in \mathsf{L} \subseteq \mathcal{C}$, we obtain $\mathcal{C} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mDEPTH}[o(\sqrt{n})]$.

Proof of 3. Suppose that $\mathsf{NP} \subseteq \mathsf{NC}^1$. By Theorem 26, there exists a monotone graph property $f \in \mathsf{NC}^1$ such that $\mathsf{mSIZE}(f) = 2^{\Omega(n^\varepsilon)}$ for some $\varepsilon > 0$. Let $\delta = \varepsilon/2$. By Lemma 12 (Item 2), there exists a monotone graph property g on $N = 2^{n^\delta}$ vertices computed by an AC^0 circuit of size $N^{2+o(1)}$ such that f is a monotone projection of g. Theorem 26 implies that any monotone circuit computing f has size $2^{\Omega(n^\varepsilon)} = 2^{\Omega((\log N)^2)} = N^{\omega(1)}$. ◀

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 4 and 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 [67, 38, 18, 19, 4] (Section 5.2). We will then prove Theorem 5 in Section 5.3, and we will finally prove Theorem 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 [18, 19]. We also refer the reader to Section 1.1.3 for the definition of the family of functions $\mathsf{CSP\text{-}SAT}_S$, as well as the terms *constraint application, S-formula* and *satisfiable formula.*

We denote by $p_i^n : \{0,1\}^n \to \{0,1\}$ the i-th *projection function* on n variables, whose operation is defined as $p_i^n(x) = x_i$. 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 $p_1^n \in [B]$ 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 $[\{\wedge, \neg\}]$), monotone functions (equal to $[\{\wedge, \vee, 0, 1\}]$), and linear functions (equal to $[\{\oplus, 1\}]$).

▶ Remark 28. The set of all clones forms a lattice, known as *Post's lattice*, under the operations $[A] \sqcap [B] := [A] \cap [B]$ and $[A] \sqcup [B] := [A \cup B]$. From the next section onwards, we will refer to the clones defined in [18] (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., $\mathrm{S}_1, \mathrm{I}_0$, etc).

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

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

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

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

We say that $\mathsf{CSP\text{-}SAT}_S$ is *trivial* if $\mathsf{CSP\text{-}SAT}_S$ is a constant function.

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

▶ **Definition 29.** *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 $\mathsf{Pol}(R)$. For a set of relations S, we denote by $\mathsf{Pol}(S)$ the set of Boolean functions which are polymorphisms of all the relations of S. For a set of Boolean functions, we denote by $\mathsf{Inv}(B)$ the set of all Boolean relations which are invariant under all functions of B (i.e., $\mathsf{Inv}(B) = \{R : B \subseteq \mathsf{Pol}(R)\}$).

The following summarises the important facts about clones, co-clones and polymorphisms that are relevant to the study of CSPs [39].

▶ **Lemma 30.** *Let S and S' be sets of Boolean relations and let B and B' be sets of Boolean functions. We have*

(i) $\mathsf{Pol}(S)$ *is a clone and* $\mathsf{Inv}(B)$ *is a co-clone;*
(ii) *If* $S \subseteq S'$, *then* $\mathsf{Pol}(S') \subseteq \mathsf{Pol}(S)$*;*
(iii) *If* $B \subseteq B'$, *then* $\mathsf{Inv}(B') \subseteq \mathsf{Inv}(B)$*;*
(iv) $\mathsf{COQ}(\mathsf{COQ}(S)) = \mathsf{COQ}(S)$*;*
(v) *If* $S \subseteq S'$, *then* $\mathsf{COQ}(S) \subseteq \mathsf{COQ}(S')$*;*
(vi) $\mathsf{Inv}(\mathsf{Pol}(S)) = \langle S \rangle$*;*
(vii) $\mathsf{Pol}(\mathsf{Inv}(B)) = [B]$.

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 $f \leq_m^{\mathsf{mOR}} g$ if there exists a many-one monotone OR-reduction from f to g. We also write $f \leq_m^{\mathsf{AC}^0} g$ if there exists a many-one AC^0 reduction from f to g, and $f \leq_m^{\mathsf{mNL}} g$ if there exists a many-one mNL reduction from f to g[15]. Unless otherwise specified, every reduction we consider will generate an instance of polynomial size on the length of the input.

Finally, we denote by OR^k and NAND^k the k-ary OR and NAND relations, respectively.

5.2 Basic facts about CSP-SAT

We state here basic facts about the $\mathsf{CSP\text{-}SAT}$ function. These facts are proved in the original paper of Schaefer [67], as well as in later papers [38, 18, 19, 4].

Lemma 31 below is one of the most important lemmas of this section and will be used many times. It states that $\mathsf{Pol}(S)$ characterises the monotone complexity of $\mathsf{CSP\text{-}SAT}_S$, 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 [38, 19, Theorem 2.4], and we check in Appendix B.2 that their proofs also hold in the monotone case.

▶ **Lemma 31** (Polymorphisms characterise the complexity of CSPs [38, 19, Theorem 2.4]). *If* $\mathsf{Pol}(S_2) \subseteq \mathsf{Pol}(S_1)$, *then* $\mathsf{CSP\text{-}SAT}^n_{S_1} \leq_m^{\mathsf{mNL}} \mathsf{CSP\text{-}SAT}^{\mathsf{poly}(n)}_{S_2}$.

Theorem 32 gives monotone circuit upper bounds for some instances of $\mathsf{CSP\text{-}SAT}_S$. Non-monotone variants of this upper bound were originally obtained in the seminal paper of Schaefer [67], and we again check that the monotone variants work in Appendix B.3.

▶ **Theorem 32** (Monotone version of the upper bounds for CSP-SAT [67, 4]). *Let S be a finite set of relations. The following holds.*

1. *If* $\mathrm{E}_2 \subseteq \mathsf{Pol}(S)$ *or* $\mathrm{V}_2 \subseteq \mathsf{Pol}(S)$, *then* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mSIZE}[\mathsf{poly}]$.
2. *If* $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$, *or* $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$, *or* $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S)$, *then* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL}$.

Finally, we state here a result of [4], which classifies the *non-monotone* complexity of $\mathsf{CSP\text{-}SAT}_S$ under $\leq_m^{\mathsf{AC}^0}$ reductions. The classification of the complexity of $\mathsf{CSP\text{-}SAT}_S$ is based solely on $\mathsf{Pol}(S)$. See Figure 1 for a graphical representation.

[15] 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 $\rho_e \in \{1, x_1, \ldots, x_n\}$. 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 $\rho_e(x) = 1$.

▶ **Theorem 33** (Refined classification of CSP problems [4, Theorem 3.1]). *Let S be a finite set of Boolean relations. The following holds.*

- *If* $\mathrm{I}_0 \subseteq \mathsf{Pol}(S)$ *or* $\mathrm{I}_1 \subseteq \mathsf{Pol}(S)$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is trivial.*
- *If* $\mathsf{Pol}(S) \in \{\mathrm{I}_2, \mathrm{N}_2\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leq_m^{\mathsf{AC}^0}$*-complete for* NP*.*
- *If* $\mathsf{Pol}(S) \in \{\mathrm{V}_2, \mathrm{E}_2\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leq_m^{\mathsf{AC}^0}$*-complete for* P*.*
- *If* $\mathsf{Pol}(S) \in \{\mathrm{L}_2, \mathrm{L}_3\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leq_m^{\mathsf{AC}^0}$*-complete for* $\oplus\mathsf{L}$*.*
- *If* $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S) \subseteq {\mathrm{S}_{00}}^2$ *or* $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S) \subseteq {\mathrm{S}_{10}}^2$ *or* $\mathsf{Pol}(S) \in \{\mathrm{D}_2, \mathrm{M}_2\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leq_m^{\mathsf{AC}^0}$*-complete for* NL*.*
- *If* $\mathsf{Pol}(S) \in \{\mathrm{D}_1, \mathrm{D}\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leq_m^{\mathsf{AC}^0}$*-complete for* L*.*
- *If* $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$ *or* $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$*, then either* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{AC}^0$ *or* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leq_m^{\mathsf{AC}^0}$*-complete for* L*.*

5.3 A monotone dichotomy for CSP-SAT

In this section, we prove Theorem 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).

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 $\mathsf{3\text{-}XOR\text{-}SAT}$ to $\mathsf{CSP\text{-}SAT}_S$.

▶ **Lemma 34.** *Let S be a finite set of relations. If* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$*, then* $\mathsf{3\text{-}XOR\text{-}SAT} \leq_m^{\mathsf{mNL}} \mathsf{CSP\text{-}SAT}_S$*.*

Proof. Inspecting Post's lattice (Figure 1), note that the only clones strictly contained in L_3 are $\mathrm{L}_2, \mathrm{N}_2$ and I_2. We will first show that the reduction holds for the case $\mathsf{Pol}(S) = \mathrm{L}_2$ and then prove that the reduction also holds for the case $\mathsf{Pol}(S) = \mathrm{L}_3$. Lemma 31 will then imply the cases $\mathsf{Pol}(S) \in \{\mathrm{N}_2, \mathrm{I}_2\}$, since $\mathrm{I}_2 \subseteq \mathrm{N}_2 \subseteq \mathrm{L}_3$.

It's not hard to check that, if $\mathsf{Pol}(S) = \mathrm{L}_2$, then $\mathsf{Pol}(S) \subseteq \mathsf{Pol}(\mathsf{3\text{-}XOR\text{-}SAT})$ (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 31 we deduce that $\mathsf{3\text{-}XOR\text{-}SAT}$ admits a reduction to $\mathsf{CSP\text{-}SAT}_S$ in mNL. In order to prove the case $\mathsf{Pol}(S) = \mathrm{L}_3$, we first prove the following claim.

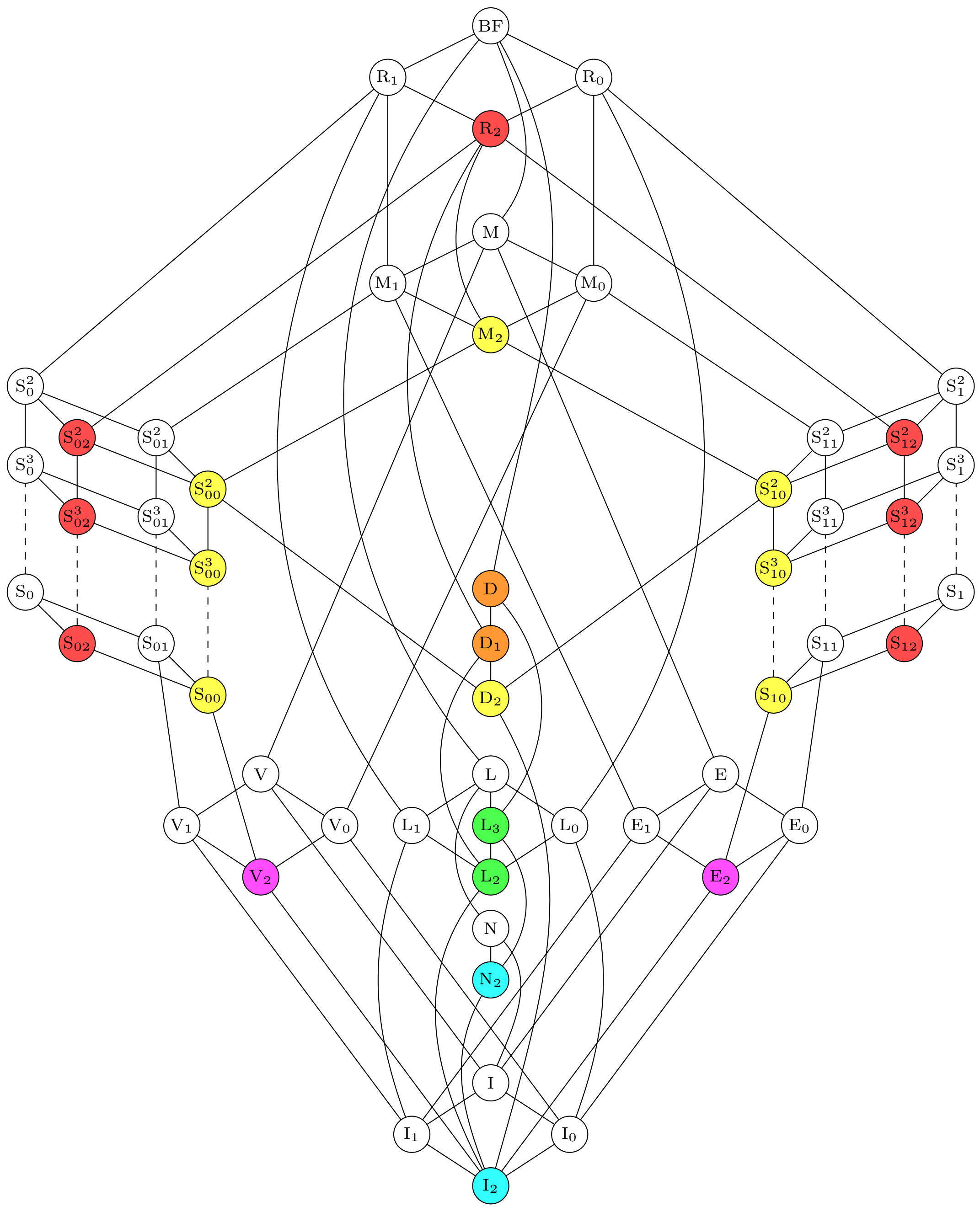
▷ Claim ([4, Lemma 3.11]). Let S be a finite set of relations such that $\mathsf{Pol}(S) = \mathrm{L}_2$. There exists a finite set of relations S' such that $\mathsf{Pol}(S') = \mathrm{L}_3$ and $\mathsf{CSP\text{-}SAT}_S^n \leq_m^{\mathsf{mProj}} \mathsf{CSP\text{-}SAT}_{S'}^{n+1}$.

Proof. We describe the proof of Lemma 3.11 in [4] and observe that it gives a monotone reduction.

For a relation $R \in S$, let $R' = \{(\neg x_1, \dots, \neg x_k) : (x_1, \dots, x_k) \in R\}$. Let also $S' = \{R' : R \in S\}$. It's not hard to check that $\mathsf{Pol}(S') = \mathrm{L}_3$, 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 $\rho \in \mathsf{Pol}(S')$ and ρ is a Boolean function on at least two bits, then $\rho \in \mathsf{Pol}(S) = \mathrm{L}_2$.

Now let F be a instance of $\mathsf{CSP\text{-}SAT}_S^n$. For every constraint $C = R(x_1, \dots, x_k)$ in F, we add the constraint $C' = R'(\alpha, x_1, \dots, x_k)$ to the S'-formula F', where α 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. ◁

Since the case $\mathsf{Pol}(S) = \mathrm{L}_2$ holds, the case $\mathsf{Pol}(S) = \mathrm{L}_3$ now follows from Lemma 31 and the Claim. Finally, from Lemma 31 we conclude that the reduction also holds for the case $\mathsf{Pol}(S) \in \{\mathrm{N}_2, \mathrm{I}_2\}$, since $\mathrm{I}_2 \subseteq \mathrm{N}_2 \subseteq \mathrm{L}_3$. ◀



🔵 NP-complete 🟣 P-complete 🟢 $\oplus$L-complete
🟡 NL-complete 🟠 L-complete 🔴 L-complete or in coNLOGTIME
⚪ Trivial

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 $\leq_m^{\mathsf{AC}^0}$ reductions. See Theorem 33 for details. A similar figure appears in [4, Figure 1].

▶ **Theorem 35** (Dichotomy for monotone circuits). *Let S be a finite set of relations. If* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$ *then there exists a constant* $\varepsilon > 0$ *such that* $\mathsf{mSIZE}(\mathsf{CSP\text{-}SAT}_S) = 2^{\Omega(n^{\varepsilon})}$. *Otherwise, we have* $\mathsf{mSIZE}(\mathsf{CSP\text{-}SAT}_S) = n^{O(1)}$.

Proof. If $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$, the lower bound follows from the 'moreover' part of Theorem 7, and Lemma 34. For the upper bound, we inspect Post's lattice (Figure 1). Observe that, if $\mathsf{Pol}(S) \not\subseteq \mathrm{L}_3$, the following are the only possible cases:

1. $\mathrm{I}_0 \subseteq \mathsf{Pol}(S)$ or $\mathrm{I}_1 \subseteq \mathsf{Pol}(S)$. In both cases, any $\mathsf{CNF}(S)$ is trivially satisfiable.
2. $\mathrm{E}_2 \subseteq \mathsf{Pol}(S)$ or $\mathrm{V}_2 \subseteq \mathsf{Pol}(S)$. In this case, $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mSIZE}[\mathsf{poly}]$ by Theorem 32.
3. $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$. In this case, $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$ by Theorem 32. ◀

▶ Remark 36. We remark that the lifting theorem of [31] (which is an ingredient in the proof of Theorem 7) is only used to prove that the monotone complexity of $\mathsf{CSP\text{-}SAT}_S$ is exponential when $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$. If we only care to show a superpolynomial separation, then it suffices to apply the superpolynomial lower bound for CSPs with counting proved in [27, 9] using the approximation method. Indeed, we give an explicit proof in Appendix A. The same holds for the consequences of this theorem (see Theorem 46).

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

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

The following is proved in [56, 31].

▶ **Theorem 37** ([56, 31]). *There exists* $\varepsilon > 0$ *such that* $\mathsf{3\text{-}Horn\text{-}SAT} \in \mathsf{mSIZE}[\mathsf{poly}] \setminus \mathsf{mDEPTH}[o(n^{\varepsilon})]$.

Proof sketch. Since $\mathrm{E}_2 \subseteq \mathsf{Pol}(\mathcal{H}^3)$ (see, e.g., [25, Lemma 4.8]), the upper bound follows from Theorem 32. The lower bound follows from a lifting theorem of [56, 31]. 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^{ε}, the lower bound follows. ◀

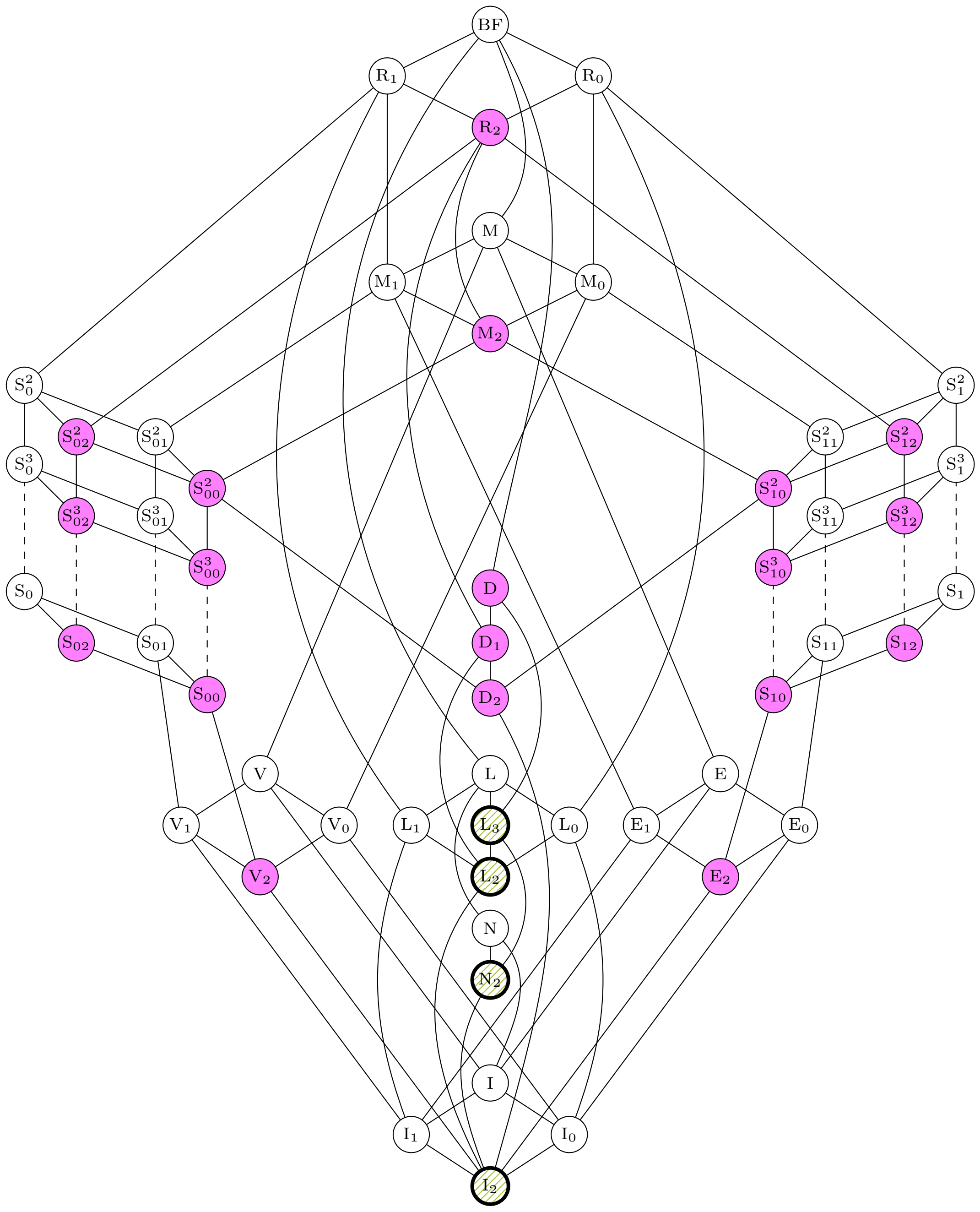
Analogously to the previous section, we show that 3-Horn-SAT reduces to $\mathsf{CSP\text{-}SAT}_S$ whenever $\mathsf{Pol}(S)$ is small enough, in a precise sense stated below. We then deduce the dichotomy for formulas with a similar argument.

▶ **Lemma 38.** *Let S be a finite set of relations. If* $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$ *or* $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$, *then* $\mathsf{3\text{-}Horn\text{-}SAT} \leq_m^{\mathsf{mNL}} \mathsf{CSP\text{-}SAT}_S$.

Proof. We first consider the case $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$. Note that $\mathrm{E}_2 \subseteq \mathsf{Pol}(\mathsf{3\text{-}Horn\text{-}SAT})$ (see, e.g., [25, Lemma 4.8]). Therefore, from Lemma 31 we deduce that 3-Horn-SAT admits a reduction to $\mathsf{CSP\text{-}SAT}_S$ in mNL.

Now let $\mathsf{3\text{-}AntiHorn\text{-}SAT} = \mathsf{CSP\text{-}SAT}_{\mathcal{A}^3}$, where $\mathcal{A}^3 = \{(x_1 \vee x_2 \vee \neg x_3), (x_1 \vee x_2 \vee x_3), (\neg x)\}$. Observe that a $\mathcal{H}^3$-formula φ is satisfiable if and only if the $\mathcal{A}^3$-formula $\varphi(\neg x_1, \dots, \neg x_n)$ is satisfiable. Therefore, we have $\mathsf{3\text{-}Horn\text{-}SAT} \leq_m^{\mathsf{mProj}} \mathsf{3\text{-}AntiHorn\text{-}SAT}$. Observing that $\mathrm{V}_2 \subseteq \mathsf{Pol}(\mathcal{A}^3)$ (again, see e.g. [25, Lemma 4.8]), the result now follows from Lemma 31 and the previous paragraph. ◀

▶ **Theorem 39** (Dichotomy for monotone formulas). *Let S be a finite set of relations. If* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$, *or* $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$, *or* $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$, *then there is a constant* $\varepsilon > 0$ *such that* $\mathsf{mDEPTH}(\mathsf{CSP\text{-}SAT}_S) = \Omega(n^{\varepsilon})$. *Otherwise, we have* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL} \subseteq \mathsf{mNC}^2 \subseteq \mathsf{mDEPTH}[\log^2 n]$.



● Solvable in mSIZE[poly].

▨ Requires monotone circuits of size $2^{\Omega(n^{\varepsilon})}$.

○ Trivial (i.e., a constant function).

Figure 2 Illustration of Theorem 35. The vertices are colored with the *monotone circuit size* complexity of deciding CSPs whose set of polymorphisms corresponds to the label of the vertex.

Proof. We will first prove the lower bound. If $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$, the lower bound follows from Theorem 35. If $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$ or $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$, the lower bound follows from Theorem 37 and Lemma 38.

By inspecting Post's lattice (Remark 28), we see that the remaning cases are:

1. $\mathrm{I}_0 \subseteq \mathsf{Pol}(S)$ or $\mathrm{I}_1 \subseteq \mathsf{Pol}(S)$. In both cases, any $\mathsf{CNF}(S)$ is trivially satisfiable.
2. $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$, or $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S)$, or $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$. In all of those three cases, we have $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL}$ by Theorem 32. ◀

5.4 Some auxiliary results

In this section, we prove auxiliary results needed in the proof of a more general form of Theorem 4. In particular, we will prove that all $\mathsf{CSP\text{-}SAT}_S$ which are in AC^0 are also contained in $\mathsf{mAC}^0 \subseteq \mathsf{mNC}^1$. Moreover, we show that, if $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mNC}^1$, then $\mathsf{CSP\text{-}SAT}_S$ is L-hard under $\leq_m^{\mathsf{AC}^0}$ reductions.

We first observe that, when $\mathsf{COQ}(S_1) \subseteq \mathsf{COQ}(S_2)$, there exists an efficient low-depth reduction from $\mathsf{CSP\text{-}SAT}_{S_1}$ to $\mathsf{CSP\text{-}SAT}_{S_2}$. This reduction, which will be useful in this section, is more refined than the one given by Lemma 31. A proof of the non-monotone version of this statement is found in [19, Proposition 2.3], and we give a monotone version of this proof in Appendix B.2.

▶ **Lemma 40** ([19, Proposition 2.3])**.** *If $\mathsf{COQ}(S_1) \subseteq \mathsf{COQ}(S_2)$, then there exists a constant $C \in \mathbb{N}$ such that $\mathsf{CSP\text{-}SAT}^n_{S_1} \leq_m^{\mathsf{mOR}} \mathsf{CSP\text{-}SAT}^{Cn}_{S_2}$.*

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

We now recall some lemmas from [4], and prove a few consequences from them. We say that a set S of relations *can express equality* if $\{=\} \subseteq \mathsf{COQ}(S)$.

▶ **Lemma 41** ([4])**.** *Let S be a finite set of relations. Suppose $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S)$ ($\mathrm{S}_{12} \subseteq \mathsf{Pol}(S)$, resp.) and that S cannot express equality. Then there exists $k \geq 2$ such that $S \subseteq \mathsf{COQ}(\left\{\mathsf{OR}^k, x, \neg x\right\})$ ($S \subseteq \mathsf{COQ}(\left\{\mathsf{NAND}^k, x, \neg x\right\})$, resp.).*

Proof. Follows from the proof of Lemma 3.8 of [4]. ◀

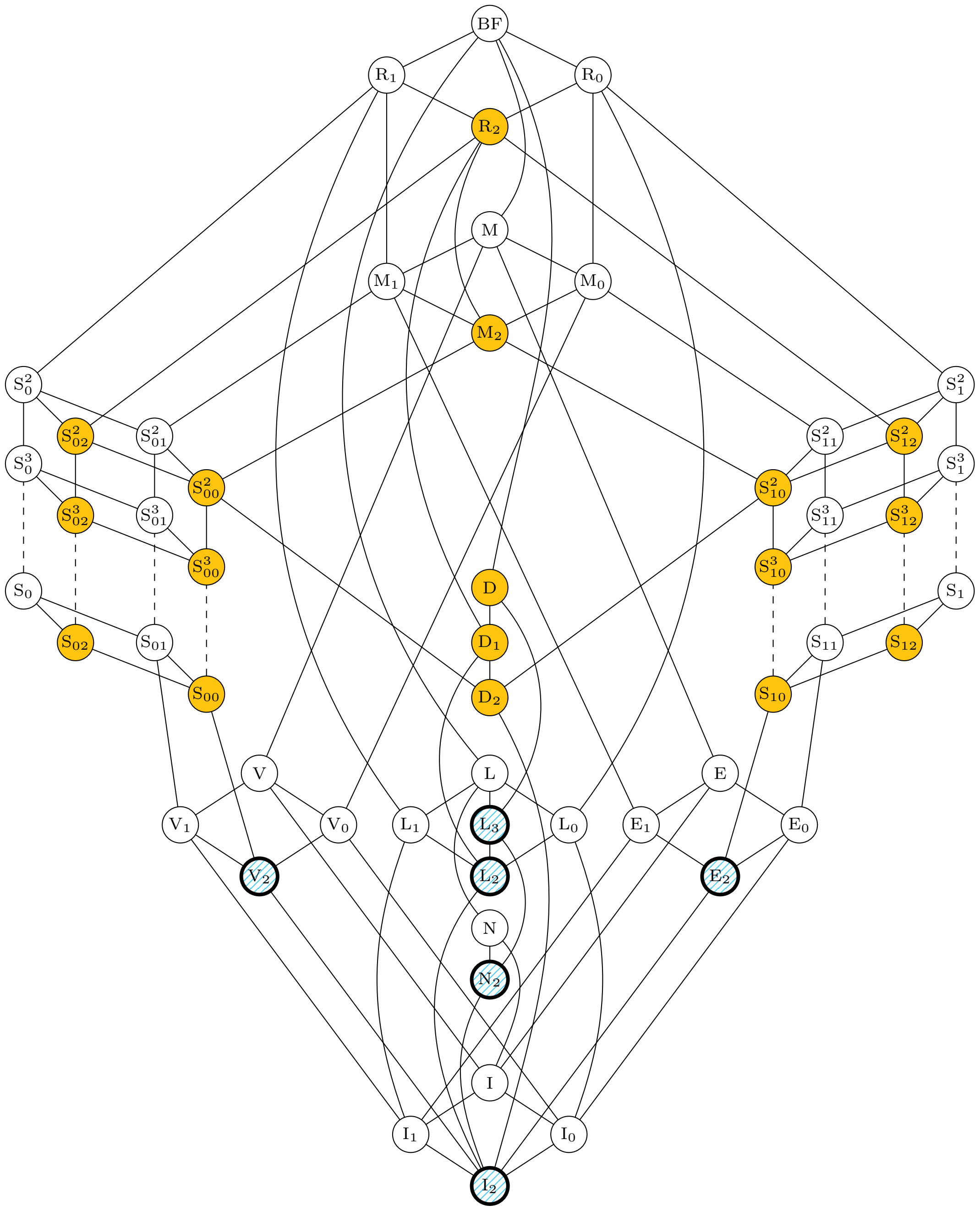
▶ **Lemma 42.** *Let S be a finite set of relations such that $\mathsf{Pol}(S) \subseteq \mathrm{R}_2$. If $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S)$ or $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S)$, and S cannot express equality, then $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mAC}^0_3$.*

Proof. We write the proof in the case $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S)$. The other case is analogous.

From Lemmas 40 and 41 and Items iv and v of Lemma 30, we get that there is a monotone OR-reduction from $\mathsf{CSP\text{-}SAT}_S$ to $\mathsf{CSP\text{-}SAT}_{\{\mathsf{OR}^k, x, \neg x\}}$ for some k. However, an $\left\{\mathsf{OR}^k, x, \neg x\right\}$-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 $\mathsf{CSP\text{-}SAT}_S$. ◀

▶ **Lemma 43** ([4, Lemma 3.8])**.** *Let S be a finite set of relations such that $\mathsf{Pol}(S) \subseteq \mathrm{R}_2$. If $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S)$ or $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S)$, and S can express equality, then $\mathsf{CSP\text{-}SAT}_S$ is L-hard under $\leq_m^{\mathsf{AC}^0}$ reductions.*

▶ **Lemma 44.** *Let S be a finite set of relations. If $\mathrm{S}_{02} \not\subseteq \mathsf{Pol}(S)$ and $\mathrm{S}_{12} \not\subseteq \mathsf{Pol}(S)$, then $\mathsf{CSP\text{-}SAT}_S$ is L-hard or trivial.*



● Solvable in $\mathsf{mNL} \subseteq \mathsf{mDEPTH}[O(\log^2 n)]$.
▨ Requires monotone depth $\Omega(n^{\varepsilon})$.
○ Trivial (i.e., a constant function).

Figure 3 Illustration of Theorem 39. The vertices are colored with the *monotone circuit depth* complexity of deciding CSPs whose set of polymorphisms corresponds to the label of the vertex.

Proof. This follows by inspecting Post's lattice (Figure 1) and the classification theorem (Theorem 33). ◀

We may now prove the main result of this subsection.

▶ **Theorem 45.** *We have* $\mathsf{CSP} \cap \mathsf{AC}^0 \subseteq \mathsf{mAC}^0_3$. *Moreover, if* $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mAC}^0_3$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* L*-hard under* $\leq_m^{\mathsf{AC}^0}$ *reductions.*

Proof. Let S be a finite set of relations. If $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mAC}^0_3$, then, by Lemma 42, at least one of the following cases hold:

1. $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$ or $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$, and S can express the equality relation;
2. $\mathrm{S}_{02} \not\subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$ and $\mathrm{S}_{12} \not\subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$.
3. $\mathsf{Pol}(S) \not\subseteq \mathrm{R}_2$.

Since $\mathsf{CSP\text{-}SAT}_S$ is not trivial, we obtain that $\mathsf{CSP\text{-}SAT}_S$ is L-hard in the first two cases by Lemmas 43 and 44, and it's easy to check that $\mathsf{CSP\text{-}SAT}_S$ is also L-hard in the third case by inspecting Post's lattice (Figure 1) and the classification theorem (Theorem 33). Since $\mathsf{L} \not\subseteq \mathsf{AC}^0$, this also implies that, if $\mathsf{CSP\text{-}SAT}_S \in \mathsf{AC}^0$, then $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$ or $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$, and S cannot express the equality relation. Lemma 42 again gives $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mAC}^0_3$. ◀

5.5 Consequences for monotone circuit lower bounds via lifting

We now prove a stronger form of Theorem 4. In the previous section, we showed that $\mathsf{CSP} \cap \mathsf{AC}^0 \subseteq \mathsf{mAC}^0$. In particular, this means that there does not exist a finite set of relations S such that $\mathsf{CSP\text{-}SAT}_S$ separates AC^0 and mNC^1, a separation which we proved in Theorem 1. We will also observe that, if $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mNC}^2$, then $\mathsf{CSP\text{-}SAT}_S$ is $\oplus\mathsf{L}$-hard.

▶ **Theorem 46.** *Let S be a finite set of Boolean relations.*

1. *If* $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mAC}^0_3$ *then* $\mathsf{CSP\text{-}SAT}_S$ *is* L*-hard under* $\leq_m^{\mathsf{AC}^0}$ *reductions.*
2. *If* $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mNC}^2$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\oplus\mathsf{L}$*-hard under* $\leq_m^{\mathsf{AC}^0}$ *reductions.*

Proof. Item (1) follows from Theorem 45. To prove item (2), suppose that $\mathsf{mDEPTH}(\mathsf{CSP\text{-}SAT}_S) = \omega(\log^2 n)$. Then, by Theorem 39, we conclude that $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$, or $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$, or $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$. Theorem 33 implies that $\mathsf{CSP\text{-}SAT}_S$ is $\oplus\mathsf{L}$-hard. ◀

Further Discussion. We recall the discussion of Section 1.1.3. We introduced and defined the functions $\mathsf{CSP\text{-}SAT}_S$ 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 [31] to require monotone circuit lower bounds of size $2^{n^{\Omega(1)}}$ to compute, even though $\oplus\mathsf{L}$-machines running in polynomial-time can compute it. Theorem 46 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 $\mathsf{CSP\text{-}SAT}_S \in \mathsf{AC}^0$, but the monotone complexity of $\mathsf{CSP\text{-}SAT}_S$ is large. However, Item (1) of Theorem 46 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 $\mathsf{AC}^0[\oplus]$ with superpolynomial monotone circuit complexity, as proved in Theorem 7. However, Item (2) of Theorem 46 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 9.

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 [67] has been extended to non-Boolean CSPs [72, 15].

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 46. One example is the Tardós function [69]. 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 $k = n^{\varepsilon}$ [56, 61]. 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 [69], there is a monotone function in P which has the same output behaviour over these instances.

References

1 Miklós Ajtai and Yuri Gurevich. Monotone versus positive. *J. Assoc. Comput. Mach.*, 34(4):1004–1015, 1987. doi:10.1145/31846.31852.

2 Miklós Ajtai, János Komlós, and Endre Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 1–9. ACM, 1983. doi:10.1145/800061.808726.

3 Jin Akiyama and Mikio Kano. *Factors and Factorizations of Graphs*, volume 2031 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-21919-1.

4 Eric Allender, Michael Bauland, Neil Immerman, Henning Schnoor, and Heribert Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *J. Comput. Syst. Sci.*, 75(4):245–254, 2009. doi:10.1016/j.jcss.2008.11.001.

5 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table. *SIAM J. Comput.*, 38(1):63–84, 2008. doi:10.1137/060664537.

6 Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 295–302. IEEE Computer Society, 2001. doi:10.1109/CCC.2001.933896.

7 Noga Alon and Ravi B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987.

8 Alexander E Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Doklady Akademii Nauk SSSR*, 282:1033–1037, 1985.

9 László Babai, Anna Gál, and Avi Wigderson. Superpolynomial lower bounds for monotone span programs. *Combinatorica*, 19(3):301–319, 1999. doi:10.1007/s004930050058.

10 Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *Advances in Cryptology* (CRYPTO), pages 27–35, 1988.

11 S. J. Berkowitz. On some relationships between monotone and non-monotone circuit complexity. Technical report, University of Toronto, 1982.

12 Eric Blais, Johan Håstad, Rocco A. Servedio, and Li-Yang Tan. On DNF approximators for monotone boolean functions. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 235–246, 2014.

13 Eric Blais, Dominik Scheder, and Li-Yang Tan. Ajtai-gurevich redux. Manuscript, 2013.

14 Nader H. Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996. doi:10.1145/234533.234564.

15 Andrei A. Bulatov. A dichotomy theorem for nonuniform csps. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.37.

16 Andrei A. Bulatov. Constraint satisfaction problems: complexity and algorithms. *ACM SIGLOG News*, 5(4):4–24, 2018. doi:10.1145/3292048.3292050.

17 Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and importance of logspace-mod class. *Math. Syst. Theory*, 25(3):223–237, 1992. doi:10.1007/BF01374526.

18 Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with boolean blocks, part i: Post's lattice with applications to complexity theory. *SIGACT News*, 2003.

19 Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with boolean blocks, part ii: Constraint satisfaction problems. *ACM SIGACT-Newsletter*, 35, 2004.

20 Bruno Pasqualotto Cavalar, Mrinal Kumar, and Benjamin Rossman. Monotone circuit lower bounds from robust sunflowers. In *LATIN 2020: Theoretical Informatics - 14th Latin American Symposium, São Paulo, Brazil, January 5-8, 2021, Proceedings*, volume 12118 of *Lecture Notes in Computer Science*, pages 311–322. Springer, 2020. doi:10.1007/978-3-030-61792-9_25.

21 Arkadev Chattopadhyay, Rajit Datta, and Partha Mukhopadhyay. Lower bounds for monotone arithmetic circuits via communication complexity. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 786–799. ACM, 2021. doi:10.1145/3406325.3451069.

22 Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *Symposium on Theory of Computing* (STOC), pages 670–683, 2016.

23 Lijie Chen, Shuichi Hirahara, Igor C. Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond Natural Proofs: Hardness Magnification and Locality. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151, pages 70:1–70:48, 2020.

24 Xi Chen, Igor C. Oliveira, and Rocco A. Servedio. Addition is exponentially harder than counting for shallow monotone circuits. In *STOC'17 – Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1232–1245. ACM, New York, 2017. doi:10.1145/3055399.3055425.

25 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of Boolean constraint satisfaction problems*, volume 7 of *SIAM monographs on discrete mathematics and applications*. SIAM, 2001.

26 Susanna F. de Rezende, Mika Göös, and Robert Robere. Guest column: Proofs, circuits, and communication. *SIGACT News*, 53(1):59–82, 2022. doi:10.1145/3532737.3532746.

27 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.

28 Oded Goldreich, Shafi Goldwasser, Eric Lehman, and Dana Ron. Testing monotonicity. In *Symposium on Foundations of Computer Science,* (FOCS), pages 426–435, 1998.

29 Oded Goldreich and Rani Izsak. Monotone circuits: One-way functions versus pseudorandom generators. *Theory Comput.*, 8(1):231–238, 2012. doi:10.4086/toc.2012.v008a010.

30 Mika Göös, Rahul Jain, and Thomas Watson. Extension complexity of independent set polytopes. *SIAM J. Comput.*, 47(1):241–269, 2018. doi:10.1137/16M109884X.

31 Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *10th Innovations in Theoretical Computer Science*, volume 124 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 38, 19. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019.

32 Michelangelo Grigni and Michael Sipser. Monotone complexity. In *Proceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, pages 57–75, USA, 1992. Cambridge University Press.

33 Siyao Guo, Tal Malkin, Igor C. Oliveira, and Alon Rosen. The power of negations in cryptography. In *Theory of Cryptography Conference* (TCC), pages 36–65, 2015.

34 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986. `doi:10.1145/12130.12132`.

35 Johan Håstad, Ingo Wegener, Norbert Wurm, and Sang-Zin Yi. Optimal depth, very small size circuits for symmetric functions in AC^0. *Inf. Comput.*, 108(2):200–211, 1994. `doi:10.1006/inco.1994.1008`.

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

37 Christian Ikenmeyer, Balagopal Komarath, Christoph Lenzen, Vladimir Lysikov, Andrey Mokhov, and Karteek Sreenivasaiah. On the complexity of hazard-free circuits. *J. ACM*, 66(4):25:1–25:20, 2019. `doi:10.1145/3320123`.

38 Peter Jeavons. On the algebraic structure of combinatorial problems. *Theor. Comput. Sci.*, 200(1-2):185–204, 1998. `doi:10.1016/S0304-3975(97)00230-2`.

39 Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, July 1997. `doi:10.1145/263867.263489`.

40 Jan Johannsen. The complexity of satisfiability problems with two occurrences, 2003. Unpublished Manuscript.

41 Neil D. Jones, Y. Edmund Lien, and William T. Laaser. New problems complete for nondeterministic log space. *Math. Syst. Theory*, 10:1–17, 1976. `doi:10.1007/BF01683259`.

42 Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers.* Springer, 2012.

43 Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discret. Math.*, 5(4):545–557, 1992. `doi:10.1137/0405044`.

44 Mauricio Karchmer and Avi Wigderson. On span programs. In *Structure in Complexity Theory Conference* (CCC), pages 102–111, 1993. `doi:10.1109/SCT.1993.336536`.

45 Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. Symbolic Logic*, 62(2):457–486, 1997.

46 Denis Kuperberg. Positive first-order logic on words. In *Symposium on Logic in Computer Science* (LICS), pages 1–13, 2021.

47 Denis Kuperberg. Positive first-order logic on words and graphs. *CoRR*, abs/2201.11619, 2022. `arXiv:2201.11619`.

48 Benoît Larose and Pascal Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410(18):1629–1647, 2009. `doi:10.1016/j.tcs.2008.12.048`.

49 E. A. Okol'nishnikova. The effect of negations on the complexity of realization of monotone Boolean functions by formulas of bounded depth. *Metody Diskret. Analiz.*, pages 74–80, 1982.

50 Igor C. Oliveira. Unconditional lower bounds in complexity theory, 2015. (PhD Thesis, Columbia University).

51 Igor C. Oliveira, Rahul Santhanam, and Srikanth Srinivasan. Parity helps to compute majority. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, pages 23:1–23:17, 2019. `doi:10.4230/LIPIcs.CCC.2019.23`.

52 Toniann Pitassi and Robert Robere. Strongly exponential lower bounds for monotone computation. In *Symposium on Theory of Computing* (STOC), pages 1246–1255, 2017. `doi:10.1145/3055399.3055478`.

53 Emil L. Post. *The Two-Valued Iterative Systems of Mathematical Logic. (AM-5)*. Princeton University Press, 1941. URL: `http://www.jstor.org/stable/j.ctt1bgzb1r`.

54 Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.*, 62(3):981–998, 1997. `doi:10.2307/2275583`.

55 W. V. Quine. Two theorems about truth functions. *Bol. Soc. Mat. Mexicana*, 10:64–70, 1953.

56 Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999. `doi:10.1007/s004930050062`.

57 Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992. `doi:10.1145/146637.146684`.

58 A. A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR*, 281(4):798–801, 1985.

59 Alexander A Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical Notes of the Academy of Sciences of the USSR*, 37(6):485–493, 1985.

60 Omer Reingold. Undirected st-connectivity in log-space. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 376–385. ACM, 2005. `doi:10.1145/1060590.1060647`.

61 Susanna Rezende. Personal communication, 2023.

62 Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential lower bounds for monotone span programs. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 406–415. IEEE Computer Society, 2016. `doi:10.1109/FOCS.2016.51`.

63 Benjamin Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3):Art. 15, 53, 2008. `doi:10.1145/1379759.1379763`.

64 Benjamin Rossman. On the constant-depth complexity of k-clique. In *Symposium on Theory of Computing* (STOC), pages 721–730, 2008.

65 Benjamin Rossman. An entropy proof of the switching lemma and tight bounds on the decision-tree size of AC^0, 2017.

66 Benjamin Rossman. An improved homomorphism preservation theorem from lower bounds in circuit complexity. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 27:1–27:17, 2017.

67 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978. `doi:10.1145/800133.804350`.

68 Alexei P. Stolboushkin. Finitely monotone properties. In *Symposium on Logic in Computer Science* (LICS), pages 324–330, 1995.

69 É. Tardos. The gap between monotone and nonmonotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988. `doi:10.1007/BF02122563`.

70 Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. `doi:10.1145/7531.8928`.

71 Leslie G. Valiant. Negation can be exponentially powerful. *Theor. Comput. Sci.*, 12:303–314, 1980.

72 Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342. IEEE Computer Society, 2017. `doi:10.1109/FOCS.2017.38`.

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

As discussed in Section 1.1.3, [31] 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 $\mathsf{OddFactor}_n : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ 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 $\mathsf{OddFactor}$, which takes as an input a *bipartite* graph with n vertices on each part, and accepts if the graph contains an odd factor. Let $\mathsf{Bipartite\text{-}OddFactor}_n : \{0,1\}^{n^2} \to \{0,1\}$ be this function. We remark that the lower bounds of Babai, Gál and Wigderson [9] for $\mathsf{OddFactor}$ (Theorem 10) also hold for $\mathsf{Bipartite\text{-}OddFactor}$. The proof of the monotone circuit lower bound in particular is essentially Razborov's lower bound for $\mathsf{Matching}$ via the approximation method [59].

▶ **Theorem 47** ([9]). *We have*

$$\mathsf{mSIZE}(\mathsf{Bipartite\text{-}OddFactor}_n) = n^{\Omega(\log n)} \quad \textit{and} \quad \mathsf{mDEPTH}(\mathsf{Bipartite\text{-}OddFactor}_n) = \Omega(n).$$

We can reduce $\mathsf{Bipartite\text{-}OddFactor}$ to $\mathsf{3\text{-}XOR\text{-}SAT}$ by noting that computing $\mathsf{Bipartite\text{-}OddFactor}_n(M)$ 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 $\mathsf{3\text{-}XOR\text{-}SAT}$ 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 $\mathsf{3\text{-}SAT}$. As the corresponding reductions turn out to be monotone, this implies monotone circuit and formula lower bounds for $\mathsf{3\text{-}XOR\text{-}SAT}$. We note that a somewhat similar argument (in the non-monotone setting) appears in Feder and Vardi [27, 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 48.** *Let f be a Boolean function. We define* $\mathsf{dual}(f) : x \mapsto \neg f(\neg x)$ *as the* dual *of f.*

▶ **Lemma 49.** *Let f be a monotone Boolean function. We have* $\mathsf{mSIZE}(f) = \mathsf{mSIZE}(\mathsf{dual}(f))$ *and* $\mathsf{mDEPTH}(f) = \mathsf{mDEPTH}(\mathsf{dual}(f))$.

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 $\{\wedge, \vee\}$-circuit computing f into a circuit computing $\mathsf{dual}(f)$ by swapping $\wedge$-gates for $\vee$-gates, and vice-versa. Moreover, this transformation preserves the depth of the circuit. ◀

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

▶ **Theorem 50.** *There exists $\varepsilon > 0$ such that*

$$\mathsf{mSIZE}(\mathsf{3\text{-}XOR\text{-}SAT}) = n^{\Omega(\log n)} \quad \textit{and} \quad \mathsf{mDEPTH}(\mathsf{3\text{-}XOR\text{-}SAT}) = \Omega(n^{\varepsilon}).$$

Proof. Recall that the value of the function $\mathsf{Bipartite\text{-}OddFactor}_n(M)$ 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 $z_{i1}, \dots, z_{i(n-1)}$ and the equations

$$\begin{aligned} z_{i1} &= x_{i1} \oplus x_{i2}, \\ z_{i2} &= z_{i1} \oplus x_{i3}, \\ &\dots \\ z_{i,(n-1)} &= z_{i,(n-2)} \oplus x_{i,n}, \\ z_{i,(n-1)} &= 1. \end{aligned}$$

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 $w_{j1}, \dots, w_{j(n-1)}$ 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 $\mathsf{Bipartite\text{-}OddFactor}_n(M) = 1$. Moreover, it is easy to see that the characteristic vector α 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 $f = \mathsf{dual}(\mathsf{3\text{-}XOR\text{-}SAT})$ and $\beta = \neg\alpha$. Since, by definition, 3-XOR-SAT accepts *unsatisfiable* systems, we get $\mathsf{Bipartite\text{-}OddFactor}_n(M) = \neg\mathsf{3\text{-}XOR\text{-}SAT}(\alpha) = f(\beta)$ and that β is a *monotone* projection of M. Therefore, by Lemma 49, we obtain

$$\mathsf{mSIZE}(\mathsf{Bipartite\text{-}OddFactor}_n) \leq \mathsf{mSIZE}(\mathsf{3\text{-}XOR\text{-}SAT})$$

and

$$\mathsf{mDEPTH}(\mathsf{Bipartite\text{-}OddFactor}_n) \leq \mathsf{mDEPTH}(\mathsf{3\text{-}XOR\text{-}SAT}).$$ ◀

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 $\mathsf{ST\text{-}CONN} : \{0,1\}^{n^2} \to \{0,1\}$ be the function that outputs 1 on a given directed graph G if there exists a path from 1 to n in G. Let $\mathsf{GEN} : \{0,1\}^{n^3} \to \{0,1\}$ 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 51** ([42, Exercise 7.3], [56]). *We have* $\mathsf{ST\text{-}CONN} \in \mathsf{mNL}$ *and* $\mathsf{GEN} \in \mathsf{mSIZE[poly]}$.

B.2 Proof of reduction lemmas

Here we present monotonised versions of the proofs of [19, Propositions 2.2 - 2.4], which give a simplified presentation of the results of [67].

▶ **Lemma 40** ([19, Proposition 2.3]). *If* $\mathsf{COQ}(S_1) \subseteq \mathsf{COQ}(S_2)$, *then there exists a constant* $C \in \mathbb{N}$ *such that* $\mathsf{CSP\text{-}SAT}^{n}_{S_1} \leq^{\mathsf{mOR}}_{m} \mathsf{CSP\text{-}SAT}^{Cn}_{S_2}$.

Proof. If $\mathsf{COQ}(S_1) \subseteq \mathsf{COQ}(S_2)$, 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 $\mathsf{CNF}(S_2)$ 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. ◀

▶ **Lemma 52.** *Let S be a set of Boolean relations. We have* $\mathsf{CSP\text{-}SAT}_{S\cup\{=\}} \leq_m^{\mathsf{mNL}} \mathsf{CSP\text{-}SAT}_S$.

Proof. Let F be a $(S \cup \{=\})$-formula on n variables given as an input. Remember that F is given as a Boolean vector α, where each bit of α represents the presence of a constraint application on n variables from $S \cup \{=\}$. We first build an undirected graph G with the variables $x_1, \dots, x_n$ 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 $C = R(x_1, \dots, x_n)$ be a constraint application of R. If C appears in F, we add to the system every constraint of the form $C' = R(y_1, \dots, y_n)$ 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 NL using the monotone NL algorithm for $\mathsf{ST\text{-}CONN}$ (Theorem 51). Indeed, there are at most n^k constraint applications of a given relation R of arity k. Therefore, to decide if a constraint $C' = R(y_1, \dots, y_n)$ 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 $\mathsf{ST\text{-}CONN}$. ◀

▶ **Lemma 31** (Polymorphisms characterise the complexity of CSPs [38, 19, Theorem 2.4]). *If* $\mathsf{Pol}(S_2) \subseteq \mathsf{Pol}(S_1)$, *then* $\mathsf{CSP\text{-}SAT}^n_{S_1} \leq_m^{\mathsf{mNL}} \mathsf{CSP\text{-}SAT}^{\mathsf{poly}(n)}_{S_2}$.

Proof. If $\mathsf{Pol}(S_2) \subseteq \mathsf{Pol}(S_1)$, then from Lemma 30 (Items iii, v, and vi) we obtain $\mathsf{COQ}(S_1) \subseteq \langle S_1 \rangle \subseteq \langle S_2 \rangle = \mathsf{COQ}(S_2 \cup \{=\})$. Therefore, by Lemmas 40 and 52 we can do the following chain of reductions in monotone NL:

$$\mathsf{CSP\text{-}SAT}_{S_1} \leq_m^{\mathsf{mOR}} \mathsf{CSP\text{-}SAT}_{S_2\cup\{=\}} \leq_m^{\mathsf{mNL}} \mathsf{CSP\text{-}SAT}_{S_2}. \quad ◀$$

B.3 Monotone circuit upper bounds

We restate and prove the theorem.

▶ **Theorem 32** (Monotone version of the upper bounds for CSP-SAT [67, 4]). *Let S be a finite set of relations. The following holds.*

1. *If* $\mathrm{E}_2 \subseteq \mathsf{Pol}(S)$ *or* $\mathrm{V}_2 \subseteq \mathsf{Pol}(S)$, *then* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mSIZE}[\mathsf{poly}]$.
2. *If* $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$, *or* $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$, *or* $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S)$, *then* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL}$.

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 $\mathsf{GEN} \in \mathsf{mSIZE}[\mathsf{poly}]$. Indeed, we interpret each constraint of the form $(\neg x_i \vee \neg x_j \vee x_k)$ (which is equivalent to $x_i \wedge x_j \implies x_k$) as a triple (i, j, k), and constraints of the form x_i as a triple $(0, 0, i)$. Let $S \subseteq \{0, 1, 2 \dots, n\}$ be the

set generated by these triples, applying the generation rules of GEN, using $0 \in S$ as the axiom. It suffices to check that there exists some constraint of the form $\neg x_i \vee \neg x_j \vee \neg x_k$, such that $\{i,j,k\} \subseteq S$. This process can be done with polynomial-size monotone circuits, invoking GEN. Therefore, it follows from Theorem 51 that 3-Horn-SAT $\in \mathsf{mSIZE}[\mathsf{poly}]$.
Moreover, we recall that, if $\mathrm{E}_2 \subseteq \mathsf{Pol}(S)$, then $S \subseteq \mathsf{COQ}(\mathcal{H}^3)$ (in other words, every S-formula can be written as a set of 3-Horn equations) – see, e.g, [25, Lemma 4.8]. Therefore, from Items iv and v of Lemma 30 and Lemma 40, we conclude that $\mathsf{CSP\text{-}SAT}_S \leq_m^{\mathsf{mOR}}$ 3-Horn-SAT $\in \mathsf{mSIZE}[\mathsf{poly}]$.
Now recall that, if $\mathrm{V}_2 \subseteq \mathsf{Pol}(S)$, then $S \subseteq \mathsf{COQ}(\mathcal{A}^3)$, where $\mathcal{A}^3$ is the set of width-3 Anti-Horn relations (i.e., $\mathcal{A}^3 = \{(x_1 \vee x_2 \vee \neg x_3), (x_1 \vee x_2 \vee x_3), (\neg x)\}$; see [25, Lemma 4.8] for a proof of this observation). But note that an $\mathcal{A}^3$-formula φ is satisfiable if and only if the $\mathcal{H}^3$-formula $\varphi(\neg x_1, \dots, \neg x_n)$ is satisfiable. Therefore by Lemma 40 and Items iv and v of Lemma 30, we have $\mathsf{CSP\text{-}SAT}_S \leq_m^{\mathsf{mOR}} \mathsf{CSP\text{-}SAT}_{\mathcal{A}^3} \leq_m^{\mathsf{mProj}}$ 3-Horn-SAT $\in \mathsf{mSIZE}[\mathsf{poly}]$.

Proof of 2. We first prove the case $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$. Let 2-SAT $= \mathsf{CSP\text{-}SAT}_\Gamma$, where $\Gamma = \{(x_1 \vee x_2), (x_1 \vee \neg x_2), (\neg x_1 \vee \neg x_2)\}$. It's easy to check that the standard reduction from 2-SAT to ST-CONN can be done in monotone NL (see [41, Theorem 4]). Therefore, it follows from Theorem 51 that 2-SAT $\in \mathsf{mNL}$. Now, recall that, if $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$, then $S \subseteq \mathsf{COQ}(\Gamma)$ (see, e.g., [25, Lemma 4.9]). Therefore, from Lemma 40 and Items iv and v of Lemma 30, we conclude $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL}$.
We now suppose that $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$. We check that the proof of [4, Lemma 3.4] gives a monotone circuit. If $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$, then there exists $k \geq 2$ such that ${\mathrm{S}_{00}}^k \subseteq \mathsf{Pol}(S)$ (that's because there does not exist a finite set of relations S such that $\mathsf{Pol}(S) = \mathrm{S}_{00}$). Note that ${\mathrm{S}_{00}}^k = \mathsf{Pol}(\Gamma)$, where $\Gamma = \left\{\mathsf{OR}^k, x, \neg x, \to, =\right\}$. We show below how to decide if a Γ-formula is unsatisfiable in monotone NL. The result then follows from Lemma 31.
Let F be a given Γ-formula with n variables. We first construct a directed graph G, with vertex set $\{x_1, \dots, x_n\}$, and with arcs (x_i, x_j) if $x_i \to x_j$ 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 Γ-formula F is unsatisfiable if, and only if, there exists a constraint of the form $x_{i_1} \vee \dots \vee x_{i_k}$ 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 NL by Theorem 51.
The case $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S)$ is analogous. ◀

C 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 $\to : (x,y) \mapsto (\neg x \vee y)$. Let also $\leftrightarrow : (x,y) \mapsto \neg(x \oplus y)$ and $\mathrm{id} : x \mapsto x$. Let $f : \{0,1\}^k \to \{0,1\}$ 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 $f(x) = \langle c, x\rangle + b \pmod 2$. We say that f is self-dual if $f = \mathsf{dual}(f)$. Let $a \in \{0,1\}$. We say that f is *a-reproducing* if $f(a,\dots,a) = a$. 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 $B = [F]$.

Name	Definition	Base
BF	All Boolean functions	$\{\vee, \wedge, \neg\}$
R_0	$\{f \in BF : f$ is 0-reproducing$\}$	$\{\wedge, \oplus\}$
R_1	$\{f \in BF : f$ is 1-reproducing$\}$	$\{\vee, \leftrightarrow\}$
R_2	$R_1 \cap R_0$	$\{\vee, x \wedge (y \leftrightarrow z)\}$
M	$\{f \in BF : f$ is monotonic$\}$	$\{\vee, \wedge, 0, 1\}$
M_1	$M \cap R_1$	$\{\vee, \wedge, 1\}$
M_0	$M \cap R_0$	$\{\vee, \wedge, 0\}$
M_2	$M \cap R_2$	$\{\vee, \wedge\}$
S_0^n	$\{f \in BF : f$ is 0-separating of degree $n\}$	$\{\rightarrow, \mathsf{dual}(h_n)\}$
S_0	$\{f \in BF : f$ is 0-separating$\}$	$\{\rightarrow\}$
S_1^n	$\{f \in BF : f$ is 1-separating of degree $n\}$	$\{x \wedge \overline{y}, h_n\}$
S_1	$\{f \in BF : f$ is 1-separating$\}$	$\{x \wedge \overline{y}\}$
S_{02}^n	$S_0^n \cap R_2$	$\{x \vee (y \wedge \overline{z}), \mathsf{dual}(h_n)\}$
S_{02}	$S_0 \cap R_2$	$\{x \vee (y \wedge \overline{z})\}$
S_{01}^n	$S_0^n \cap M$	$\{\mathsf{dual}(h_n), 1\}$
S_{01}	$S_0 \cap M$	$\{x \vee (y \wedge z), 1\}$
S_{00}^n	$S_0^n \cap R_2 \cap M$	$\{x \vee (y \wedge z), \mathsf{dual}(h_n)\}$
S_{00}	$S_0 \cap R_2 \cap M$	$\{x \vee (y \wedge z)\}$
S_{12}^n	$S_1^n \cap R_2$	$\{x \wedge (y \vee \overline{z}), h_n\}$
S_{12}	$S_1 \cap R_2$	$\{x \wedge (y \vee \overline{z})\}$
S_{11}^n	$S_1^n \cap M$	$\{h_n, 0\}$
S_{11}	$S_1 \cap M$	$\{x \wedge (y \vee z), 0\}$
S_{10}^n	$S_1^n \cap R_2 \cap M$	$\{x \wedge (y \vee z), h_n\}$
S_{10}	$S_1 \cap R_2 \cap M$	$\{x \wedge (y \vee z)\}$
D	$\{f \in BF : f$ is self-dual$\}$	$\{(x \wedge \overline{y}) \vee (x \wedge \overline{z}) \vee (\overline{y} \wedge \overline{z})\}$
D_1	$D \cap R_2$	$\{(x \wedge y) \vee (x \wedge \overline{z}) \vee (y \wedge \overline{z})\}$
D_2	$D \cap M$	$\{(x \wedge y) \vee (y \wedge z) \vee (x \wedge z)\}$
L	$\{f \in BF : f$ is linear$\}$	$\{\oplus, 1\}$
L_0	$L \cap R_0$	$\{\oplus\}$
L_1	$L \cap R_1$	$\{\leftrightarrow\}$
L_2	$L \cap R$	$\{x \oplus y \oplus z\}$
L_3	$L \cap D$	$\{x \oplus y \oplus z \oplus 1\}$
V	$\{f \in BF : f$ is constant or an n-ary OR function$\}$	$\{\vee, 0, 1\}$
V_0	$[\{\vee\}] \cup [\{0\}]$	$\{\vee, 0\}$
V_1	$[\{\vee\}] \cup [\{1\}]$	$\{\vee, 1\}$
V_2	$[\{\vee\}]$	$\{\vee\}$
E	$\{f \in BF : f$ is constant or an n-ary AND function$\}$	$\{\wedge, 0, 1\}$
E_0	$[\{\wedge\}] \cup [\{0\}]$	$\{\wedge, 0\}$
E_1	$[\{\wedge\}] \cup [\{1\}]$	$\{\wedge, 1\}$
E_2	$[\{\wedge\}]$	$\{\wedge\}$
N	$[\{\neg\}] \cup [\{0\}] \cup [\{1\}]$	$\{\neg, 1\}$
N_2	$[\{\neg\}]$	$\{\neg\}$
I	$[\{\mathrm{id}\}] \cup [\{0\}] \cup [\{1\}]$	$\{\mathrm{id}, 0, 1\}$
I_0	$[\{\mathrm{id}\}] \cup [\{0\}]$	$\{\mathrm{id}, 0\}$
I_1	$[\{\mathrm{id}\}] \cup [\{1\}]$	$\{\mathrm{id}, 1\}$
I_2	$[\{\mathrm{id}\}]$	$\{\mathrm{id}\}$

Figure 4 Table of all closed classes of Boolean functions, and their bases. Here, h_n denotes the function $h_n(x_1, \dots, x_{n+1}) = \bigvee_{i=1}^{n+1} \bigwedge_{j=1, j \neq i}^{n+1} x_j$. See Definition 48 for the definition of $\mathsf{dual}(\cdot)$. The same table appears in [4, Table 1].