Electronic Colloquium on Computational Complexity, Report No. 103 (2012)

Testing Assignments of Boolean CSPs

Arnab Bhattacharyya[*] Yuichi Yoshida[†]

Abstract

Given an instance $\mathcal{I}$ of a CSP, a tester for $\mathcal{I}$ distinguishes assignments satisfying $\mathcal{I}$ from those which are far from any assignment satisfying $\mathcal{I}$. The efficiency of a tester is measured by its query complexity, the number of variable assignments queried by the algorithm. In this paper, we characterize the hardness of testing Boolean CSPs according to the relations used to form constraints. In terms of computational complexity, we show that if a Boolean CSP is sublinear-query testable (resp., not sublinear-query testable), then the CSP is in NL (resp., P-complete, $\oplus$L-complete or NP-complete) and that if a sublinear-query testable Boolean CSP is constant-query testable (resp., not constant-query testable), then counting the number of solutions of the CSP is in P (resp., #P-complete). The classification is done by showing an $\Omega(n)$ lower bound for testing Horn 3-SAT and investigating Post's lattice, the inclusion structure of Boolean algebras associated with CSPs.

1 Introduction

In *property testing*, we want to decide whether an instance satisfies some particular property or is far from the property. More specifically, an algorithm is called an ε*-tester* for a property if, given an instance, it accepts with probability at least 2/3 if the instance satisfies the property, and it rejects with probability at least 2/3 if the instance is ε-far from the property. Here, an instance is called ε*-far* from a property if we must modify an ε-fraction of the instance to make it satisfy the property. The concept of property testing was introduced in [23] and extended to a combinatorial setting in [14]. Since then, many problems have been revealed to be testable in constant time, that is, independent of input size. See [13, 21, 22] for surveys to overview this area.

In *constraint satisfaction problems* (*CSP*s), we are given a set of variables and a set of constraints imposed on variables. The objective is to find an assignment that satisfies all the constraints. Depending on the relations used to make constraints, CSPs coincide with many fundamental problems such as SAT, graph coloring and linear equation systems. Of course, every Boolean property of n-bit strings can be characterized as the property of satisfying some Boolean CSP on n variables.

In this paper, we are concerned with testing whether a given assignment satisfies a particular CSP instance. That is, for a known instance $\mathcal{I}$ of a CSP, we want to distinguish assignments which satisfy $\mathcal{I}$ from those which are ε-far from satisfying $\mathcal{I}$. In this context, an assignment α on n variables is said to be ε*-far* from satisfying $\mathcal{I}$ if α differs on at least εn variables from any assignment that satisfies $\mathcal{I}$.

The efficiency of a tester for CSP assignments is measured by its *query complexity*, that is, the number of variable assignments queried by the testing algorithm. We investigate the following question:

[*]Princeton University & Center for Computational Intractability. Email: abhatt@mit.edu
[†]National Institute of Informatics and Preferred Infrastructure, Inc. Email: yyoshida@nii.ac.jp

ISSN 1433-8092

Is the query complexity of testing assignments characterized by the structure of the constraints used to form the CSP instance?

In what follows, instead of saying testing assignments for a CSP instance $\mathcal{I}$, we usually simply say testing $\mathcal{I}$. By query complexity of a class $\mathcal{C}$ of CSPs, we mean the worst case query complexity, over all CSP instances $\mathcal{I}$ in $\mathcal{C}$, of testing $\mathcal{I}$.

Testing Boolean CSPs has been already studied. For example, in [2], it was shown that testing 3-LIN and 3-SAT require $\Omega(n)$ queries, where n is the number of variables. Note that although 3-LIN is in P while 3-SAT is a classic NP-complete problem, they behave similarly in terms of query complexity. Using a universal algebraic argument, now a basic tool to study CSPs [5, 15], Yoshida showed that testing any NP-complete Boolean[1] CSP requires $\Omega(n)$ queries [25].

Due to the seminal result by Schaefer [24], the satisfiability of a Boolean CSP is polynomial-time solvable if and only if it is (equivalent to) either one of a 0-valid CSP, a 1-valid CSP, 2-SAT, Horn SAT, Dual Horn SAT, or system of linear equations (see Section 2 for definitions). Since testing any NP-complete Boolean CSP requires $\Omega(n)$ queries by [25], it is natural to ask which polynomial-time solvable CSPs are testable with a sublinear number of queries. The difficulty of testing a 0-valid or 1-valid CSP is determined by the CSP after making it non 0-valid and 1-valid somehow. Also, [12] showed that monotonicity of a function over a poset is testable with $O(\sqrt{n})$ queries, and this result implies that 2-SAT is testable with $O(\sqrt{n})$ queries. Thus, prior to this work, the only remaining Boolean CSPs whose query complexity to test was unknown were Horn SAT and Dual Horn SAT.

Our Results. The first contribution of this paper is giving an $\Omega(n)$ lower bound for testing Horn k-SAT for $k \geqslant 3$. Here, Horn k-SAT is a special case of Horn SAT, in which each constraint has arity at most k. (If $k = 2$, then Horn k-SAT becomes testable with $O(\sqrt{n})$ queries since it becomes a special case of 2-SAT.) The proof is via an interesting reduction from testing 3-LIN, which uses the fact that the hard instances of 3-LIN in [2] have expanders as their underlying graph.

The second and main contribution of this paper is a classification of Boolean CSPs with respect to their query complexity. Let $\mathcal{A} = \langle A; \Gamma \rangle$ be a pair of a domain and a set of relations (called a *relational structure*), and let CSP($\mathcal{A}$) denote the CSP that can use relations in Γ to make constraints. Then, for Boolean structures $\mathcal{A}$, we give necessary and sufficient conditions on $\mathcal{A}$ for each of the following three cases: (i) CSP($\mathcal{A}$) is constant-query testable, (ii) CSP($\mathcal{A}$) is sublinear-query testable but not constant-query testable, and (iii) CSP($\mathcal{A}$) is not sublinear-query testable.

Let $\mathcal{A}' = (A; \Gamma')$, where Γ' is any set of relations that can be made from Γ by using existential quantifiers, conjunctions and equality relations. It is known that CSP($\mathcal{A}'$) has a log-space reduction [6, 15] to CSP($\mathcal{A}$) [5, 15]. Similarly, it is known that we can test CSP($\mathcal{A}'$) with query complexity that almost matches the one for testing CSP($\mathcal{A}$) [25]. A useful tool to study CSP($\mathcal{A}'$) for $\mathcal{A}'$ made this way is universal algebra, and we can associate an "algebra" for each such $\mathcal{A}'$ (see Section 4 for details). Algebras on Boolean domain are already completely identified, and its inclusion structure is known as Post's lattice [20] (see Figure 1 and Table B). Since the missing piece in Boolean CSPs was Horn SAT and we now have a linear lower bound, we can investigate each algebra in Post's lattice and give a complete classification.

We can summarize our classification using its connection to computational complexity.

Theorem 1.1. *If a Boolean CSP is sublinear-query testable (resp., not sublinear-query testable), then the CSP is in* NL *(resp.,* P*-complete,* $\oplus$L*-complete or* NP*-complete). If a sublinear-query testable Boolean CSP is constant-query testable (resp., not constant-query testable), then counting the number of solutions of the CSP is in* P *(resp.,* #P*-complete).*

[1] In fact, [25] also implies the same result for non-Boolean CSP's if the Dichotomy Conjecture is true.

We actually give upper and lower bounds on the query complexity for each algebra in Post's lattice. The connection to computational complexity is coincidentally obtained by combining our results and [1, 8], which studied computational complexities of algebras in Post's lattice.

The work most relevant to this paper is [25], which studied the query complexity to test List H-homomorphism. For two graphs G and H, a function $f : V(G) \to V(H)$ is called a *homomorphism* from G to H if $(f(u), f(v)) \in E(H)$ whenever $(u, v) \in E(G)$. Testing List H-homomorphism is a problem, in which given a graph G, a list constraint $L : V(G) \to 2^{V(H)}$, and a function f, we want to test whether f is a homomorphism from G to H and $f(v) \in L(v)$ for each $v \in V(G)$. Testing List H-homomorphism is a special case of testing CSPs. Similarly to our classification, [25] showed the following. List H-homomorphism is sublinear-query testable (resp., not testable) if it is in NL (resp., P-complete, $\oplus$L-complete, or NP-complete). If sublinear-query testable List H-homomorphism is constant-query testable (resp., not testable), then counting the number of solutions is in P (resp., #P-complete).

Lower bounds in [25] are shown for the problem where ε-farness is measured with respect to an arbitrary distribution of variables. Since our proof is based on [25], our lower bounds in Theorem 1.1 hold only when arbitrary distributions are allowed. However, we note that our linear lower bound for Horn 3-SAT holds even if the distribution is uniform.

Though testing Horn k-SAT requires $\Omega(n)$ queries in general, it is interesting to see whether we can break the barrier in a restricted setting. In this direction, we show that Horn k-SAT is testable with $O(\sqrt{n/\varepsilon} \cdot (k + \log n))$ queries when all variables are assigned equal weight and the underlying graph is a tree. That is, any variable can appear at most once as a positive literal (see Section A for details). Our algorithm is obtained by extending the "pairing" argument used to analyze monotonicity testing [9, 12, 17].

We remark that the problem of testing CSP assignments belongs to the massively parametrized model (see, e.g., [7, 11, 18]), where the tester is given free access to part of the input and oracle access to the rest. Here, a CSP instances $\mathcal{I}$ corresponds to the former one and an assignment f corresponds to the latter one.

Organization. In Section 2, we introduce definitions used throughout the paper. Section 3 is devoted to the linear lower bound for testing Horn k-SAT where $k \geqslant 3$. In Section 4, we classify Boolean relational structures $\mathcal{A}$ with respect to the query complexity to test CSP($\mathcal{A}$). The proof of Theorem 1.1 is in Section 4.4. The algorithm for uniformly weighted tree Horn k-SAT instances with query complexity $\tilde{O}(\sqrt{n/\varepsilon} \cdot (k + \log n))$ appears in Appendix A.

2 Preliminaries

For an integer $k \geqslant 1$, a *k-ary relation* on a domain A is a subset of A^k. A *constraint language* on a domain A is a finite set of relations on A. A *(finite) relational structure*, or simply a *structure* $\mathcal{A} = \langle A; \Gamma \rangle$ consists of a non-empty set A, called the *domain*, and a constraint language Γ on A. For a structure $\mathcal{A} = \langle A; \Gamma \rangle$, we define the problem CSP($\mathcal{A}$) as follows. An instance $\mathcal{I} = (V, \mathcal{C})$ consists of a set of variables V and a set of constraints $\mathcal{C}$. Here, each constraint $C \in \mathcal{C}$ is of the form $(v_1, \ldots, v_k; R)$, where $v_1, \ldots, v_k \in V$ are variables, R is a relation in $\mathcal{A}$ and k is the arity of R. Then, the objective is to find an assignment $f : V \to A$ that satisfies all the constraints, that is $(f(v_1), \ldots, f(v_k)) \in R$ for every constraint $C = (v_1, \ldots, v_k; R) \in \mathcal{C}$. Throughout this paper, we study Boolean CSPs, and so, A is fixed to be $\{0, 1\}$. So, we often write CSP(Γ) instead of CSP($\mathcal{A}$).

Let us formally define the CSPs that most concern us here.

– k-LIN corresponds to $\mathrm{CSP}(\Gamma_{\mathrm{LIN}})$ where $\Gamma_{\mathrm{LIN}} = \{R_0, R_1\}$, $R_0 = \{(x_1, \ldots, x_k) \mid x_1 + \cdots + x_k = 0 \pmod 2\}$ and $R_1 = \{(x_1, \ldots, x_k) \mid x_1 + \cdots + x_k = 1 \pmod 2\}$.

– k-SAT corresponds to $\mathrm{CSP}(\Gamma_{\mathrm{SAT}})$ where $\Gamma_{\mathrm{SAT}} = \{R_\phi \mid \phi \in \{0,1\}^k\}$, $R_\phi = \{0,1\}^k \setminus \{\phi\}$.

– Horn k-SAT corresponds to $\mathrm{CSP}(\Gamma_{\mathrm{Horn}})$ where $\Gamma_{\mathrm{Horn}} = \{U, R_{1^k}, R_{1^{k-1}0}\}$, $U = \{1\}$ and R_{1^k}, $R_{1^{k-1}0}$ as above. Dual Horn k-SAT corresponds to $\mathrm{CSP}(\Gamma_{\mathrm{DualHorn}})$ where $\Gamma_{\mathrm{DualHorn}} = \{Z, R_{0^k}, R_{0^{k-1}1}\}$, $Z = \{0\}$, and $R_{0^k}, R_{0^{k-1}1}$ as above.

– A CSP is said to be 0-*valid* if for every instance $\mathcal{I}$ of the CSP, the all-zero assignment satisfies $\mathcal{I}$. Similarly, a CSP is said to be 1-*valid* if the all-ones assignment satisfies every instance.

As we will describe in Section 4, there is an explicitly known collection $\mathcal{P}$ of CSPs, such that any Boolean $\mathrm{CSP}(\Gamma)$ is log-space reducible to a CSP in $\mathcal{P}$.

Let $\mathcal{I} = (V, \mathcal{C})$ be a CSP instance and $\mathbf{w} : V \to \mathbb{R}$ be a *weight function* satisfying $\sum_{v \in V} \mathbf{w}(v) = 1$. For two assignments $f, f' : V \to \{0,1\}$, we define $\mathrm{dist}_{\mathbf{w}}(f, f') = \mathbf{Pr}_v[f(v) \neq f'(v)]$, where v is chosen according to the probability distribution given by $\mathbf{w}$. We define $\mathrm{dist}_{\mathcal{I},\mathbf{w}}(f)$ as the distance of f from satisfying assignments, that is $\mathrm{dist}_{\mathcal{I},\mathbf{w}}(f) = \min_{f'} \mathrm{dist}_{\mathbf{w}}(f, f')$, where f' is over satisfying assignments of $\mathcal{I}$. We say that f is ε-far from satisfying assignments if $\mathrm{dist}_{\mathcal{I},\mathbf{w}}(f) \geqslant \varepsilon$. If $\mathbf{w}$ is the uniform distribution, then we often omit $\mathbf{w}$ in the notation.

An algorithm is called an $(\varepsilon, \eta_+, \eta_-)$-*tester* for a property P if it accepts an input with probability at least $1 - \eta_+$ when it satisfies P, and it rejects an input with probability at least $1 - \eta_-$ when it is ε-far from P. An $(\varepsilon, 1/3, 1/3)$-tester is simply referred to as an ε-*tester*. (As long as $\eta_+, \eta_- < \frac{1}{2}$, an $(\varepsilon, \eta_+, \eta_-)$-tester can be converted into an ε-tester by repeating the original tester a constant number of times and taking the majority decision.)

We always use the symbol n (resp., m) to denote the number of variables (resp., constraints) in the instance we are concerned with. For a CSP instance $\mathcal{I}$ and weight function $\mathbf{w}$, an algorithm is called an ε-*tester for* $\mathcal{I}$ if, given an assignment f for $\mathcal{I}$, it ε-tests whether f is a satisfying assignment of $\mathcal{I}$, where farness is measured using the distance function $\mathrm{dist}_{\mathcal{I},\mathbf{w}}(\cdot)$. Given a structure $\mathcal{A}$, we say that $\mathrm{CSP}(\mathcal{A})$ is *testable with query complexity* $q(n, m, \varepsilon)$ if for every instance $\mathcal{I}$ in $\mathrm{CSP}(\mathcal{A})$ and for every weight function $\mathbf{w}$, there is an ε-tester for $\mathcal{I}$ making $q(n, m, \varepsilon)$ queries.

3 Linear Lower Bound for Horn 3-SAT

In this section, we prove the following theorem.

Theorem 3.1. *There exist constants $\varepsilon, \delta, \eta \in (0,1)$ such that for all large enough n, there is a Horn 3-SAT formula $\mathcal{I}_{\mathrm{Horn}}$ on n variables and $O(n)$ constraints such that any adaptive $(\varepsilon, \eta_+, \eta_-)$-test for $\mathcal{I}_{\mathrm{Horn}}$ makes at least δn queries if $\eta_+ + \eta_- < \eta$.*

Thus, we obtain a linear lower bound for ε-testers as well. Note that Theorem 3.1 also implies a linear lower bound for ε-testing Dual Horn 3-SAT. We can simply negate each literal in the hard Horn 3-SAT formula to obtain the hard Dual Horn 3-SAT formula.

The proof of Theorem 3.1 is by a reduction from 3-LIN which is known to require $\Omega(n)$ queries. We first revisit the construction of the hard 3-LIN instance $\mathcal{I}_{\mathrm{LIN}}$. Then, we show how to reduce to Horn 3-SAT using the structure of $\mathcal{I}_{\mathrm{LIN}}$.

3.1 Construction of hard 3-LIN instance

In [2], Ben-Sasson, Harsha and Raskhodnikova constructed a 3-LIN instance $\mathcal{I}_{\mathrm{LIN}}$ such that any two-sided adaptive tester for $\mathcal{I}_{\mathrm{LIN}}$ requires $\Omega(n)$ queries. The construction proceeded in two steps.

The first step shows the existence of hard k-LIN formulae for sufficiently large k. Call a bipartite multigraph $G = (L, R, E)$ (c, k)*-regular* if the degree of every left vertex $u \in L$ is c and the degree of every right vertex $v \in R$ is k. Every (c, k)-regular graph G describes a k-LIN formula $\psi(G)$: for every right vertex $v \in R$, $\psi(G)$ contains a constraint $\sum_{u \in N(v)} x_u = 0 \pmod 2$ where $N(v)$ is the set of neighbors of v. A *random* (c, k)*-regular LDPC code of length* n is obtained by taking $\psi(G)$ for a random (c, k)-regular graph G with n left vertices. The following was shown in [2]:

Theorem 3.2 (Theorem 3.7 of [2])**.** *For any odd integer $c \geqslant 7$ and for $\mu, \varepsilon, \delta, k > 0$ satisfying:*

$$\mu \leqslant \frac{1}{100c^2}; \qquad \delta < \mu^c; \qquad k > \frac{2\mu c^2}{(\mu^c - \delta)^2}; \qquad \varepsilon \leqslant \frac{1}{100k^2}$$

and for sufficiently large n, it is the case that with high probability for a random (c, k)-regular LDPC code $\psi(G)$ of length n, every adaptive $(\varepsilon, \eta_+, \eta_-)$-test for $\psi(G)$ makes at least δn queries, if $\eta_+ + \eta_- \leqslant 1 - 2\mu$.

An important fact about random (c, k)-regular graphs that was used in the proof of Theorem 3.2 and will be useful to us is:

Lemma 3.3 (Lemma 6.3 of [2])**.** *For all integers $c \geqslant 7$, $k \geqslant 2$, and sufficiently large n, a random (c, k)-regular graph $G = (L, R, E)$ with n left vertices has the following property with high probability: for every nonempty subset $S \subseteq L$ of left vertices such that $|S| \leqslant \frac{n}{100k^2}$, there exists a right vertex $v \in R$ such that v has exactly one neighbor in S.*

In the second step, testing satisfiability of k-LIN instances is reduced to testing satisfiability of 3-LIN instances. This is done by repeating $\lceil \log(k-2) \rceil$ times a reduction $\mathcal{R}$ from k-LIN instances ψ to $(\lceil k/2 \rceil + 1)$-LIN instances $\mathcal{R}(\psi)$. If ψ is a k-LIN instance with n variables and m linear constraints $A_1, \ldots, A_m$, then $\mathcal{R}(\psi)$ is a $(\lceil k/2 \rceil + 1)$-LIN instance with $n + m$ variables and $2m$ linear constraints $A'_1, A''_1, \ldots, A'_m, A''_m$, where if the constraint A_i is $x_1 + x_2 + \cdots + x_k = 0 \pmod 2$, then the constraints A'_i and A''_i are, respectively:

$$x_1 + \cdots + x_{\lceil k/2 \rceil} + z_i = 0 \pmod 2 \quad \text{and} \quad x_{\lceil k/2 \rceil + 1} + \cdots + x_k + z_i = 0 \pmod 2$$

with z_i being a new variable.

The desired 3-LIN instance $\mathcal{I}_{\mathrm{LIN}}$ is constructed by applying the reduction $\mathcal{R}$ $\lceil \log(k-2) \rceil$ many times on a random (c, k)-regular LDPC code $\psi(G)$ with the following parameters:

$$c = 7; \qquad k = 16c^2(100c^2)^{2c-1}$$

If G has n_0 left vertices and m_0 right vertices, then $\mathcal{I}_{\mathrm{LIN}}$ has $n \leqslant (2c+1)n_0$ variables and $m \leqslant 2km_0$ constraints. The instance $\mathcal{I}_{\mathrm{LIN}}$ also has the following property, which is implicit in the proof of Lemma 3.8 in [2] but which will be convenient for us to make explicit.

Lemma 3.4 (Unique neighbor property)**.** *Suppose $\mathcal{I}_{\mathrm{LIN}}$, an instance of 3-LIN with n variables, is constructed as described above. Then, with high probability, for every nonempty subset S of variables such that $|S| \leqslant \frac{n}{300ck^2}$, there exists a constraint in $\mathcal{I}_{\mathrm{LIN}}$ which involves exactly one variable of S.*

Proof. Suppose ψ is a linear formula with n' variables $x_1, \ldots, x_{n'}$ and m' linear constraints $A_1, \ldots, A_{m'}$ such that for every subset S of variables such that $|S| \leqslant \varepsilon n'$, there exists a constraint in ψ involving exactly one variable of S. Then, we claim that also for $\mathcal{R}(\psi)$, a linear formula on $n' + m'$ variables $x_1, \ldots, x_{n'}, z_1, \ldots, z_{m'}$, it holds that for every nonempty subset T of variables such that $|T| \leqslant \varepsilon n'$, there exists a constraint in $\mathcal{R}(\psi)$ involving exactly one variable of T.

This claim is enough to prove the lemma, because $\mathcal{I}_{\text{LIN}}$ is formed by composing $\mathcal{R}$ several times on a random (c,k)-regular LDPC code $\psi(G)$. If $\mathcal{I}_{\text{LIN}}$ is on n variables, then $\psi(G)$ is on at least $\frac{n}{2c+1} \geqslant \frac{n}{3c}$ variables. For $\psi(G)$, Lemma 3.3 shows that with high probability, if S is a subset of variables of size at most $\frac{n/3c}{100k^2} = \frac{n}{300ck^2}$, then there is a constraint in $\psi(G)$ which involves exactly one variable of S. The lemma immediately follows from the claim.

It remains to prove the claim. Let T be a subset of the $n' + m'$ variables of $\mathcal{R}(\psi)$ such that $|T| \leqslant \varepsilon n'$. Let $T_0 = T \cap \{x_1, \ldots, x_{n'}\}$, the subset of T which corresponds to variables in the original formula ψ. Of course, $|T_0| \leqslant \varepsilon n'$, and so, there must exist a constraint A_i in ψ containing exactly one variable from T_0. In $\mathcal{R}(\psi)$, corresponding to A_i, there are two constraints A_i' and A_i''. Call A_i' the constraint which contains exactly one variable from T_0. Then A_i'' does not involve any variables from T_0. Now, there are two cases. If T does not contain z_i, then A_i' contains exactly one variable from T. Else, if T does contain z_i, then A_i'' contains exactly one variable from T, z_i itself. Therefore, in either case, our claim is proved. □

Ben-Sasson et al. [2] further showed that the reduction $\mathcal{R}$ preserves the query complexity of the instances. Without elaborating on their proof, we state their main result.

Theorem 3.5 (Theorem 3.1 of [2])**.** *Suppose $\mathcal{I}_{\text{LIN}}$, an instance of 3-LIN with n variables and $\Theta(n)$ constraints, is constructed as described above. Then, there exist $\varepsilon, \delta, \eta \in (0,1)$ such that with high probability over the construction of $\mathcal{I}_{\text{LIN}}$, any adaptive $(\varepsilon, \eta_+, \eta_-)$-test for $\mathcal{I}_{\text{LIN}}$ makes at least δn queries, if $\eta_+ + \eta_- \leqslant \eta$. In particular, there exists a 3-LIN formula on n variables which requires $\Omega(n)$ queries for testing satisfiability.*

3.2 Reduction to Horn 3-SAT

Let $\mathcal{I}_{\text{LIN}}$ on n variables be defined as above. We now construct an instance of Horn 3-SAT, $\mathcal{I}_{\text{Horn}}$, in the following way. For each variable x_i in $\mathcal{I}_{\text{LIN}}$, we have two variables v_i and v_i' in $\mathcal{I}_{\text{LIN}}$. For each linear constraint $x_i + x_j + x_k = 0 \pmod 2$ in $\mathcal{I}_{\text{LIN}}$, we have 12 Horn constraints in $\mathcal{I}_{\text{Horn}}$:

$$\begin{array}{lll}
v_i \wedge v_j \to v_k' & v_i \wedge v_k \to v_j' & v_j \wedge v_k \to v_i' \\
v_i' \wedge v_j \to v_k & v_i' \wedge v_k \to v_j & v_j' \wedge v_k \to v_i \\
v_i \wedge v_j' \to v_k & v_i \wedge v_k' \to v_j & v_j \wedge v_k' \to v_i \\
v_i' \wedge v_j' \to v_k' & v_i' \wedge v_k' \to v_j' & v_j' \wedge v_k' \to v_i'
\end{array}$$

Given an assignment f_{LIN} for $\mathcal{I}_{\text{LIN}}$, let the assignment f_{Horn} for $\mathcal{I}_{\text{Horn}}$ be defined as: $\forall i \in [n], f_{\text{Horn}}(v_i) = f_{\text{LIN}}(x_i), f_{\text{Horn}}(v_i') = \overline{f_{\text{LIN}}}(x_i)$

Lemma 3.6. *There exists $\varepsilon > 0$ such that for large enough n:*

(a) If f_{LIN} satisfies $\mathcal{I}_{\text{LIN}}$, then f_{Horn} also satisfies $\mathcal{I}_{\text{Horn}}$.

(b) If f_{LIN} is ε-far from satisfying $\mathcal{I}_{\text{LIN}}$, then f_{Horn} is also ε-far from satisfying $\mathcal{I}_{\text{Horn}}$.

Proof. The first part is immediate. It is easy to check that if f_{LIN} satisfies a constraint in $\mathcal{I}_{\text{LIN}}$, then f_{Horn} also satisfies the corresponding constraint in $\mathcal{I}_{\text{Horn}}$.

To see part (b), assume it is false so that f_{LIN} is ε-far from satisfying $\mathcal{I}_{\text{LIN}}$ but f_{Horn} is ε-close to a satisfying assignment g_{Horn} for $\mathcal{I}_{\text{Horn}}$. Let $S = \{x_i \mid i \in [n] \text{ such that } g_{\text{Horn}}(v_i) = g_{\text{Horn}}(v'_i)\}$. Clearly, $|S| \leqslant 2\varepsilon n$, since $f_{\text{Horn}}(v_i) \neq f_{\text{Horn}}(v'_i)$ for every i.

Also, we can prove $S \neq \emptyset$. Suppose otherwise. Then, define an assignment g_{LIN} for $\mathcal{I}_{\text{LIN}}$ as: $g_{\text{LIN}}(x_i) = g_{\text{Horn}}(v_i)$ for every $i \in [n]$. g_{LIN} is ε-close to f_{LIN}. We now show that g_{LIN} satisfies $\mathcal{I}_{\text{LIN}}$, and so f_{LIN} is ε-close to $\mathcal{I}_{\text{LIN}}$, a contradiction. Consider a constraint $x_i + x_j + x_k = 0 \pmod 2$ in $\mathcal{I}_{\text{LIN}}$. We know that $g_{\text{Horn}}(v_i) = g_{\text{LIN}}(x_i)$ and, since $|S| = 0$, $g_{\text{Horn}}(v'_i) = \overline{g_{\text{LIN}}}(x_i)$. Since g_{Horn} satisfies $\mathcal{I}_{\text{Horn}}$, the following constraints must be true:

$$\begin{gathered}
g_{\text{LIN}}(x_i) \wedge g_{\text{LIN}}(x_j) \to \overline{g_{\text{LIN}}}(x_k) \\
\overline{g_{\text{LIN}}}(x_i) \wedge g_{\text{LIN}}(x_j) \to g_{\text{LIN}}(x_k) \\
g_{\text{LIN}}(x_i) \wedge \overline{g_{\text{LIN}}}(x_j) \to g_{\text{LIN}}(x_k) \\
\overline{g_{\text{LIN}}}(x_i) \wedge \overline{g_{\text{LIN}}}(x_j) \to \overline{g_{\text{LIN}}}(x_k)
\end{gathered}$$

It is now easy to check that these constraints hold iff $g_{\text{LIN}}(x_i) + g_{\text{LIN}}(x_j) + g_{\text{LIN}}(x_k) = 0 \pmod 2$.

Therefore, $0 < |S| \leqslant 2\varepsilon n$. If ε is sufficiently small, Lemma 3.4 shows that there must exist a constraint in $\mathcal{I}_{\text{LIN}}$ that contains exactly one variable of S. On the other hand, we now show that any constraint in $\mathcal{I}_{\text{LIN}}$ containing one variable in S must contain at least one other variable in S, thus causing a contradiction and finishing the proof. Consider a constraint $x_i + x_j + x_k = 0 \pmod 2$ in $\mathcal{I}_{\text{LIN}}$, and suppose $x_i \in S$. There are two cases.

– Suppose $g_{\text{Horn}}(v_i) = g_{\text{Horn}}(v'_i) = 1$. Since g_{Horn} is a satisfying assignment, it must be:

$$\begin{aligned}
&g_{\text{Horn}}(v_j) \to g_{\text{Horn}}(v'_k) && g_{\text{Horn}}(v_k) \to g_{\text{Horn}}(v'_j) \\
&g_{\text{Horn}}(v_j) \to g_{\text{Horn}}(v_k) && g_{\text{Horn}}(v_k) \to g_{\text{Horn}}(v_j) \\
&g_{\text{Horn}}(v'_j) \to g_{\text{Horn}}(v_k) && g_{\text{Horn}}(v'_k) \to g_{\text{Horn}}(v_j) \\
&g_{\text{Horn}}(v'_j) \to g_{\text{Horn}}(v'_k) && g_{\text{Horn}}(v'_k) \to g_{\text{Horn}}(v'_j)
\end{aligned}$$

So, $g_{\text{Horn}}(v_j) = g_{\text{Horn}}(v'_j) = g_{\text{Horn}}(v_k) = g_{\text{Horn}}(v'_k)$, and therefore, $x_j, x_k \in S$.

– Suppose $g_{\text{Horn}}(v_i) = g_{\text{Horn}}(v'_i) = 0$. Then, the first eight Horn constraints corresponding to $x_i + x_j + x_k = 0$ are vacuously satisfied. The remaining four are satisfied exactly when

$$\begin{gathered}
g_{\text{Horn}}(v_j) \wedge g_{\text{Horn}}(v_k) \\
g_{\text{Horn}}(v'_j) \wedge g_{\text{Horn}}(v_k) \\
g_{\text{Horn}}(v_j) \wedge g_{\text{Horn}}(v'_k) \\
g_{\text{Horn}}(v'_j) \wedge g_{\text{Horn}}(v'_k)
\end{gathered}$$

are all false. This can only hold when $g_{\text{Horn}}(v_j) = g_{\text{Horn}}(v'_j) = 0$ or $g_{\text{Horn}}(v_k) = g_{\text{Horn}}(v'_k) = 0$. Thus, either x_j or x_k is in S.

$\square$

Theorem 3.1 now immediately follows from Theorem 3.5.

4 Classification of Boolean CSPs

In this section, we classify (finite) Boolean structures $\mathcal{A}$ into three categories with respect to the query complexity for testing CSP($\mathcal{A}$). Namely, we give necessary and sufficient conditions for each of the following three cases: (i) CSP($\mathcal{A}$) is constant-query testable, (ii) CSP($\mathcal{A}$) is sublinear-query testable but not constant-query testable, and (iii) CSP($\mathcal{A}$) is not sublinear-query testable.

4.1 Universal Algebra Preliminaries

An *n-ary operation* on a set A is a map from A^n to A. An n-ary operation f on A *preserves* the k-ary relation R on A (equivalently, we say that R is *invariant* under f) if the following holds: given any matrix M of size $k \times n$ whose columns are in R, applying f to the rows of M will produce a k-tuple in R. For instance, one can check that every binary Boolean relation is preserved by the ternary majority operation and that the ternary Boolean relation $x \wedge y \to z$ is preserved by the binary AND operation.

Given a constraint language Γ, let $\mathrm{Pol}(\Gamma)$ denote the set of all operations that preserve all relations in Γ. It's known [15] that if for two constraint languages Γ_1, Γ_2, we have $\mathrm{Pol}(\Gamma_1) = \mathrm{Pol}(\Gamma_2)$, the computational complexity of CSP(Γ_1) and CSP(Γ_2) are equal (upto log-space reducibility), and as we will soon see, their query complexities are roughly equal also. Hence, we can together study the complexity of all CSP(Γ) such that $\mathrm{Pol}(\Gamma)$ equals some particular F.

To this end, given a constraint language Γ, define $\mathrm{Alg}(\Gamma)$ to be the algebra $\langle A; \mathrm{Pol}(\Gamma)\rangle$, where the domain A of the algebra in our case is fixed to be $\{0, 1\}$. It can be easily seen that for any Γ, $\mathrm{Pol}(\Gamma)$ forms a *clone*, i.e., a set of operations closed under compositions and containing all the projections (operations of the form $f(x_1, \ldots, x_k) = x_i$). Also, in fact, the converse is true: every clone can be characterized as $\mathrm{Pol}(\Gamma)$ for some set of relations Γ [19]. Hence, it suffices to only consider algebras $\langle A; F\rangle$, where F is a clone.

Remarkably, it turns out that there is an explicit description [20] of the countably many Boolean clones. When ordered by inclusion, they form a lattice known as *Post's lattice*, shown in Figure 1 and Table B with standard notation for the operations. In the rest of this section, we will settle the query complexity for the CSPs associated to each clone in Post's lattice.

In Section 4.2, we review known upper and lower bounds. In Section 4.3, we give a classification of $\mathcal{A}$ assuming that $\mathcal{A}$ is neither 0-valid nor 1-valid. We deal with structures that are 0-valid or 1-valid in Section 4.4.

4.2 Known results

For an algebra $\mathbb{A} = (A; F)$, we say CSP($\mathbb{A}$) is *testable* with $q(n, m, \varepsilon)$ queries if, for any constraint language Γ satisfying $\mathrm{Pol}(\Gamma) = F$, CSP(Γ) is testable with $q(n, m, \varepsilon)$ queries. The following lemma states that essentially $\mathrm{Pol}(\Gamma)$ decides the query complexity for testing CSP(Γ).

Lemma 4.1 ([25]). *Given constraint language Γ, suppose* CSP(Γ) *is testable with $q(n, m, \varepsilon)$ queries.*

- *If Γ' is a constraint language such that $\mathrm{Pol}(\Gamma') \supseteq \mathrm{Pol}(\Gamma)$, then* CSP($\Gamma'$) *is testable with $O(1/\varepsilon) + q(O(n+m), O(m), O(\varepsilon))$ queries.*
- CSP($\mathrm{Alg}(\Gamma)$) *is testable with $O(1/\varepsilon) + q(O(n+m), O(m), O(\varepsilon))$ queries.*

A k-ary operation $f(x_1, \ldots, x_k)$ is called a *near-unanimity* if $f(y, x, \ldots, x) = f(x, y, x, \ldots, x) = \cdots = f(x, \ldots, x, y) = x$ for any x, y. A 3-ary near-unanimity operation is called a *majority*. In [25], it is shown that CSP($\mathbb{A}$) is testable with $O(\sqrt{n})$ queries if $\mathbb{A}$ contains a majority as its term operation. The argument can be generalized to near-unanimity operations, and thus we have the following. (We give a proof in Appendix B for completeness.)

Lemma 4.2. *Let $\mathbb{A}$ be an algebra containing a $(k+1)$-ary near-unanimity operation. Then,* $\mathrm{CSP}(\mathbb{A})$ *is testable with $O(n^{1-\frac{1}{k}})$ queries.*

Ben-Sasson et al. [2] showed a linear lower bound for 3-LIN.

Corollary 4.3. *For $R = \{(x, y, z) \mid x + y + z = 0 \pmod 2\}$, testing* $\mathrm{CSP}(\{R\})$ *requires $\Omega(n)$ queries even when $m = O(n)$.*

Also, we have the following sublinear lower bound for CSPs related to monotonicity of functions.

Lemma 4.4 ([12])**.** *Testing* $\mathrm{CSP}(\{\to\}), \mathrm{CSP}(\{\vee\})$ *and* $\mathrm{CSP}(\{\wedge\})$ *requires $\Omega\left(\frac{\log n}{\log\log n}\right)$ queries even when $m = n^{1+O(1/\log\log n)}$.*

4.3 Classification of structures that are neither 0-valid nor 1-valid

The goal of this section is showing the following classification. We use below standard notation for the clones in Post's lattice; see Table B for their definitions.

Theorem 4.5. *Let $\mathcal{A} = \langle\{0, 1\}; \Gamma\rangle$ be a structure that is neither 0-valid nor 1-valid.*

- *If $\mathrm{Pol}(\Gamma) \in \{D_1, D, R_2\}$, then $\mathrm{CSP}(\mathcal{A})$ is testable with $O(1)$ queries.*
- *If $S_{00} \subseteq \mathrm{Pol}(\Gamma) \subseteq S_{02}^2, S_{10} \subseteq \mathrm{Pol}(\Gamma) \subseteq S_{12}^2$ or $\mathrm{Pol}(\Gamma) \in \{D_2, M_2\}$, then testing $\mathrm{CSP}(\mathcal{A})$ requires $\Omega(\log n/\log\log n)$ queries and is testable with $o(n)$ queries. The lower bound holds even when $m = n^{1+O(1/\log\log n)}$.*
- *If $\mathrm{Pol}(\Gamma) \in \{I_2, N_2, E_2, V_2, L_2, L_3\}$, then testing $\mathrm{CSP}(\mathcal{A})$ requires $\Omega(n)$ queries. The lower bound holds even when $m = O(n)$.*

If $\mathcal{A}$ is neither 0-valid nor 1-valid, then $\mathrm{Pol}(\Gamma)$ does not contain constant relations. Thus, Theorem 4.5 covers all cases (Figure 1). Lemmas 4.6, 4.7, 4.8 and 4.9 below imply Theorem 4.5.

Lemma 4.6. *If $\mathrm{Pol}(\Gamma) \in \{D_1, D, R_2\}$, then $\mathrm{CSP}(\Gamma)$ is testable with $O(1)$ queries.*

Proof. For $R = x \wedge (y \oplus z)$, we have $\mathrm{Pol}(\{R\}) = D_1$. Thus from Lemma 4.1, it suffices to show that $\mathrm{CSP}(\{R\})$ is testable with $O(1)$ queries. However, the problem is just 2-Colorability plus constant relations, and the problem is known to be testable with $O(1)$ queries (see Lemma 3.8 of [25]). □

Lemma 4.7. *If a constraint language Γ satisfies $S_{00} \subseteq \mathrm{Pol}(\Gamma) \subseteq S_{02}^2, S_{10} \subseteq \mathrm{Pol}(\Gamma) \subseteq S_{12}^2$ or $\mathrm{Pol}(\Gamma) \in \{D_2, M_2\}$, then we can test $\mathrm{CSP}(\Gamma)$ with $o(n)$ queries.*

Proof. If $\mathrm{Pol}(\Gamma) \in \{D_2, M_2\}$, the algebra contains a majority operation, and then, Lemma 4.2 shows $O(\sqrt{n})$ query complexity. Otherwise, $\mathrm{Pol}(\Gamma) \in \{S_{00}^k, S_{10}^k, S_{02}^k, S_{12}^k\}$ for some finite $k \geqslant 2$ since we assume that each relation in Γ has finite arity. In any case, $\mathrm{Pol}(\Gamma)$ contains the $(k+1)$-ary near-unanimity operation h_k. Thus, we can test $\mathrm{CSP}(\Gamma)$ with $O(n^{1-1/k})$ queries from Lemma 4.2. □

Lemma 4.8. *If a constraint language Γ satisfies $S_{00} \subseteq \mathrm{Pol}(\Gamma) \subseteq S_{02}^2, S_{10} \subseteq \mathrm{Pol}(\Gamma) \subseteq S_{12}^2$ or $\mathrm{Pol}(\Gamma) \in \{D_2, M_2\}$, then testing $\mathrm{CSP}(\Gamma)$ requires $\Omega(\frac{\log n}{\log\log n})$ queries.*

Proof. Suppose $\mathrm{Pol}(\Gamma) = M_2$, and assume that we can test $\mathrm{CSP}(\Gamma)$ with $o(\frac{\log n}{\log\log n})$ queries. Since the relation $(\rightarrow)$ is invariant under M_2, we have a tester for $\mathrm{CSP}(\{\rightarrow\})$ with $o(\frac{\log(n+m)}{\log\log n})$ queries from Lemma 4.1. However, we have a lower bound of $\Omega(\frac{\log n}{\log\log n})$ even when $m = n^{1+O(1/\log\log n)}$ from Lemma 4.4, contradiction.

We have the same lower bound for the cases $\mathrm{Pol}(\Gamma) = S_{02}^2$ and $\mathrm{Pol}(\Gamma) = S_{12}^2$. Note that the relations $(\vee)$ and $(\wedge)$ are invariant under any operation in S_{02}^2 and S_{12}^2, respectively. Thus, we have lower bounds of $\Omega(\frac{\log n}{\log\log n})$ for testing $\mathrm{CSP}(\{\vee\})$ and $\mathrm{CSP}(\{\wedge\})$ even when $m = n^{1+O(\frac{1}{\log\log n})}$ from Lemma 4.4. The same lower bound hold also for other cases from Lemma 4.1. $\square$

Lemma 4.9. *If a constraint language* Γ *satisfies* $\mathrm{Pol}(\Gamma) \in \{I_2, N_2, E_2, V_2, L_2, L_3\}$*, then testing* $\mathrm{CSP}(\Gamma)$ *requires* $\Omega(n)$ *queries.*

Proof. Note that algebras corresponding to Horn 3-SAT and Dual Horn 3-SAT are E_2 and V_2, respectively. Since we have $\Omega(n)$ lower bounds for these CSPs even when $m = O(n)$, we have the desired lower bound also for the case $\mathrm{Pol}(\Gamma) = E_2$ and $\mathrm{Pol}(\Gamma) = V_2$. From Lemma 4.1, the same lower bounds hold also for I_2.

Suppose that $\mathrm{Pol}(\Gamma) = L_2$, and we can test $\mathrm{CSP}(\Gamma)$ with $o(n)$ queries. We note that $R = \{(x, y, z) \mid x + y + z = 0 \pmod 2\}$ in Lemma 4.3 satisfies $\mathrm{Pol}(\{R\}) = L_2$. Then from Lemma 4.1, we have a tester for $\mathrm{CSP}(\{R\})$ with $o(n + m)$ queries. However, we have a lower bound of $\Omega(n)$ even when $m = O(n)$ from Corollary 4.3, contradiction.

To show the lower bound for L_3, we reduce from testing $\mathrm{CSP}(\{R\})$. Consider an instance $\mathcal{I}$ of $\mathrm{CSP}(\{R\})$ on variables $\{x_1, x_2, \ldots, x_n\}$ and a weight function $\mathbf{w}$. Let $R' = \{(x, y, z, w) : x + y + z + w = 1 \pmod 2\}$. Note that $\mathrm{Pol}(\{R'\}) = L_3$. We reduce testing $\mathcal{I}$ with respect to $\mathbf{w}$ to testing an instance $\mathcal{I}'$ of $\mathrm{CSP}(\{R'\})$ with respect to a different weight function $\mathbf{w}'$. The instance $\mathcal{I}'$ is defined on the variable set $\{t, x_1, x_2, \ldots, x_n\}$ as follows: for each equation $x_1 + x_2 + x_3 = 1$ in $\mathcal{I}$, the equation $x_1 + x_2 + x_3 + t = 1$ is contained in $\mathcal{I}'$. The weight function $\mathbf{w}'$ is set to be $\mathbf{w}'(t) = 1/2$ and $\mathbf{w}'(x_i) = \mathbf{w}(x_i)/2$ for all $i \in [n]$. Given an assignment f for $\mathcal{I}$, consider the following assignment f' for $\mathcal{I}'$: $f'(t) = 0$ and $f'(x_i) = f(x_i)$ for all $i \in [n]$. Clearly, if f satisfies $\mathcal{I}$, then f' satisfies $\mathcal{I}'$. On the other hand, for $\varepsilon < 1/2$, if f is ε-far from $\mathcal{I}$ with respect to $\mathbf{w}$, then f' is $\varepsilon/2$-far from $\mathcal{I}'$ with respect to $\mathbf{w}'$. (To see this, just note that if f' is $\varepsilon/2$-close to a satisfying assignment g', then $g'(t) = 0$ because of the weight on t.) This gives a reduction from testing $\mathrm{CSP}(\{R\})$ to testing $\mathrm{CSP}(\{R'\})$, and so, $\mathrm{CSP}(\{R'\})$ requires $\Omega(n)$ queries even if $m = O(n)$. By the same argument as in the previous paragraph, Lemma 4.1 then implies a lower bound of $\Omega(n)$ for testing $\mathrm{CSP}(\Gamma)$ for every Γ such that $\mathrm{Pol}(\Gamma) = L_3$. Additionally, Lemma 4.1 shows that the same lower bound holds also for N_2. $\square$

4.4 Classification of all structures

In this section, we examine how adding equality and constant relations affect the query complexity of CSPs. Since adding such relations make a Boolean structure neither 0-valid nor 1-valid, we can use the results of the previous section to classify the query complexity of all Boolean CSPs.

Theorem 4.10. *Given a constraint language* Γ*, suppose* Γ' *is obtained from* Γ *by adding the equality relation and constant relations.*

- *If* $\mathrm{Pol}(\Gamma') \in \{D_1, D, R_2\}$*, then* $\mathrm{CSP}(\Gamma)$ *is testable with* $O(1)$ *queries.*
- *If* $S_{00} \subseteq \mathrm{Pol}(\Gamma') \subseteq S_{02}^2, S_{10} \subseteq \mathrm{Pol}(\Gamma') \subseteq S_{12}^2$ *or* $\mathrm{Pol}(\Gamma') \in \{D_2, M_2\}$*, then testing* $\mathrm{CSP}(\Gamma)$ *requires* $\Omega(\frac{\log n}{\log\log n})$ *queries and can be done with* $o(n)$ *queries. The lower bound holds even when* $m = n^{1+O(\frac{1}{\log\log n})}$.

– *If* $\mathrm{Pol}(\Gamma') \in \{I_2, N_2, E_2, V_2, L_2, L_3\}$*, then testing* $\mathrm{CSP}(\Gamma)$ *requires* $\Omega(n)$ *queries. The lower bound holds even when* $m = O(n)$*.*

Before we prove Theorem 4.10, let us indicate how it implies Theorem 1.1 in the introduction.

Proof of Theorem 1.1. The classification of sublinear-query testable CSPs in terms of complexity of the decision problem follows directly from Theorem 4.10 and the main result of Allender et al. [1]. The classification of constant-query testable CSPs in terms of counting complexity follows from the result by Creignou and Hermann [8] that $\#\mathrm{CSP}(\Gamma)$ is in P if all relations in Γ are affine, and otherwise, $\#\mathrm{CSP}(\Gamma)$ is $\#\mathsf{P}$-complete. □

We now turn to the proof of Theorem 4.10. The following reduction is already implicitly used in Section 3.

Definition 4.11. *Given constraint languages* Γ, Γ'*, a* gap-preserving local reduction from $\mathrm{CSP}(\Gamma')$ to $\mathrm{CSP}(\Gamma)$ *exists if there are functions* $t_1(n, m), t_2(n, m)$ *and constants* c_1, c_2 *satisfying the following: given an instance* $\mathcal{I}' = (V', \mathcal{C}')$ *of* $\mathrm{CSP}(\Gamma')$*, a weight function* $\mathbf{w}'$ *and an assignment* f' *for* $\mathcal{I}'$*, there exist an instance* $\mathcal{I} = (V, \mathcal{C})$ *of* $\mathrm{CSP}(\Gamma)$*, a weight function* $\mathbf{w}$ *and an assignment* f *for* $\mathcal{I}$ *such that:*

1. $|V| \leqslant t_1(|V'|, |\mathcal{C}'|)$*.*
2. $|\mathcal{C}| \leqslant t_2(|V'|, |\mathcal{C}'|)$*.*
3. *if* f' *satisfies* $\mathcal{I}'$*, then* f *also satisfies* $\mathcal{I}$*.*
4. *if* $\mathrm{dist}_{\mathcal{I}',\mathbf{w}'}(f') \geqslant \varepsilon$*, then* $\mathrm{dist}_{\mathcal{I},\mathbf{w}}(f') \geqslant c_1\varepsilon$*.*
5. *we can compute* $f(v)$ *for any* $v \in V$ *by querying* f' *at most* c_2 *times.*

Lemma 4.12 ([25])**.** *For constraint languages* Γ, Γ'*, if there exists an* ε*-tester for* $\mathrm{CSP}(\Gamma)$ *with query complexity* $q(n, m, \varepsilon)$ *and a gap-preserving local reduction from* $\mathrm{CSP}(\Gamma')$ *to* $\mathrm{CSP}(\Gamma)$*, then there exists an* ε*-tester for* $\mathrm{CSP}(\Gamma')$ *with query complexity* $O(q(t_1(n, m), t_2(n, m), O(\varepsilon)))$*.*

To show Theorem 4.10, we need the following lemma that states that adding equality relations and constant relations does not change much the difficulty of testing CSPs.

Lemma 4.13. *Given a constraint language* Γ*, let* Γ' *be obtained from* Γ *by adding the equality relation and constant relations. Assume that* $\varepsilon \ll 1/2$*.*

– *If* $\mathrm{CSP}(\Gamma')$ *is testable with* $q(n, m, \varepsilon)$ *queries, then* $\mathrm{CSP}(\Gamma)$ *is testable with* $q(n, m, \varepsilon)$ *queries.*

– *If* $\mathrm{CSP}(\Gamma)$ *is testable with* $q(n, m, \varepsilon)$ *queries, then* $\mathrm{CSP}(\Gamma')$ *is testable with* $O(1/\varepsilon) + q(O(n + m), O(m), O(\varepsilon))$ *queries.*

Proof. The first claim is trivial. We turn to the second claim. Suppose that we can test $\mathrm{CSP}(\Gamma)$ with $q(n, m, \varepsilon)$ queries. Let $\Gamma^{=}$ be the language obtained from Γ by adding equality constraints. Since $\mathrm{Pol}(\Gamma^{=}) = \mathrm{Pol}(\Gamma)$, we can test $\mathrm{CSP}(\Gamma^{=})$ with $O(1/\varepsilon) + q(O(n + m), O(m), O(\varepsilon))$ from Lemma 4.1.

Suppose that, given an instance $\mathcal{I}' = (V', \mathcal{C}')$ of $\mathrm{CSP}(\Gamma')$, a weight function $\mathbf{w}'$ and an assignment f' for $\mathcal{I}'$, we want to test whether f' is a satisfying assignment or ε'-far from satisfying assignments. We create an instance $\mathcal{I} = (V, \mathcal{C})$ of $\mathrm{CSP}(\Gamma^{=})$, a weight function $\mathbf{w}$ and an assignment f for $\mathcal{I}$ as follows. We set $V = V' \cup \{x_0, x_1\}$ and

$$f(v) = \begin{cases} f'(v) & \text{if } v \in V' \\ i & \text{if } v = x_i, \quad i \in \{0, 1\}, \end{cases} \qquad \mathbf{w}(v) = \begin{cases} (1 - 2\varepsilon')\mathbf{w}'(v) & \text{if } v \in V' \\ \varepsilon' & \text{if } v = x_0 \text{ or } v = x_1 \end{cases}$$

Then, for each constraint of the form $(v = 0)$ in $\mathcal{C}'$, where $v \in V'$, we add a constraint of the form $(x_0 = v)$ in $\mathcal{C}$. And similarly for each constraint $(v = 1)$ in $\mathcal{C}'$, we add a constraint $(x_1 = v)$ in $\mathcal{C}$. Other constraints in $\mathcal{C}'$ are just copied to $\mathcal{C}$.

We check that the construction above is a gap-preserving local reductions from CSP(Γ') to CSP($\Gamma^=$). It is easy to see that $n = n' + 2, m = m'$ and we can take $c_2 = 1$. Also, if f' is a satisfying assignment, then f is also a satisfying assignment.

Let $\varepsilon = \varepsilon'/2$ and suppose f is ε-close to a satisfying assignment $\widetilde{f}$. Then, $\widetilde{f}(x_0) = 0$ and $\widetilde{f}(x_1) = 1$ since weights of x_0 and x_1 are larger than ε. So, $\widetilde{f}|_{V'}$ must be a valid assignment for $\mathcal{I}'$. It is easy to check that the distance between f' and $\widetilde{f}|_{V'}$ in $\mathcal{I}'$ is at most $\varepsilon/(1 - 2\varepsilon') < \varepsilon'$ when ε' is sufficiently small. Thus, we can take c_1 as $1/2$, and we can apply Lemma 4.12. □

Proof of Theorem 4.10. It is easy to see that upper bounds hold from Theorem 4.5 and the first item of Lemma 4.13.

We now see the lower bounds. Suppose Γ' satisfies the second condition and CSP(Γ) is testable with $o(\frac{\log n}{\log\log n})$ queries. Then, from the second item of Lemma 4.13, we can test CSP(Γ') with $o(\frac{\log(n+m)}{\log\log n})$ queries as well. However, we have the lower bound of $\Omega\left(\frac{\log n}{\log\log n}\right)$ for testing CSP(Γ') from Theorem 4.5 even when $m = n^{1+O(\frac{1}{\log\log n})}$, contradiction.

Suppose Γ' satisfies the third condition and CSP(Γ) is testable with $o(n)$ queries. Then, from the second item of Lemma 4.13, we can test CSP(Γ') with $o(n + m)$ queries as well. However, we have the lower bound of $\Omega(n)$ for testing CSP(Γ') from Theorem 4.5 even when $m = n$, contradiction. □

5 Conclusion

In this work, we characterized the Boolean CSPs that are testable in constant and sublinear queries, according to their defining constraint language. Besides the obvious problem for non-Boolean CSPs, here are two other interesting open questions:

- Can we classify the query complexity of testing Boolean CSPs, when the weight function is fixed to be the uniform distribution? This question is of particular interest as the unweighted Hamming distance is the most standard notion of distance.

- Can we classify the query complexity of testing *conservative* Boolean CSPs? In a conservative CSP, the set of values for each individual variable can be restricted arbitrarily. Bulatov [4] obtained a dichotomy theorem that characterized the conservative CSPs solvable in polynomial time. As for query complexity, Yoshida [25] obtained a classification of constant-query and sublinear-query testable List H-coloring problems, a particular subset of conservative CSPs.

References

[1] E. Allender, M. Bauland, N. Immerman, H. Schnoor, and H. Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *J. Comp. Sys. Sci.*, 75(4):245–254, June 2009. 3, 11

[2] E. Ben-Sasson, P. Harsha, and S. Raskhodnikova. Some 3CNF properties are hard to test. *SIAM J. on Comput.*, 35(1):1–21, 2006. 2, 5, 6, 9

[3] A. Bulatov. Combinatorial problems raised from 2-semilattices. *Journal of Algebra*, 298(2):321–339, 2006. 18

[4] A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Logic*, 12(4):1–66, July 2011. 12

[5] A. Bulatov, A. Krokhin, and P. Jeavons. Constraint satisfaction problems and finite algebras. *Proc. 27th Annual International Conference on Automata, Languages, and Programming*, pages 272–282, 2000. 2

[6] A. Bulatov and M. Valeriote. Recent results on the algebraic approach to the csp. *Complexity of Constraints*, pages 68–92, 2008. 2

[7] S. Chakraborty, E. Fischer, O. Lachish, A. Matsliah, and I. Newman. Testing st-connectivity. *Proc. 11th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 380–394, 2007. 3

[8] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Inform. and Comput.*, 125(1):1–12, Feb. 1996. 3, 11

[9] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. *Combinatorica*, 20(3):301–337, 2000. 3

[10] T. Feder and M. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. on Comput.*, 28(1):57–104, 1998. 18

[11] E. Fischer, O. Lachish, I. Newman, A. Matsliah, and O. Yahalom. On the query complexity of testing orientations for being eulerian. *Proc. 12th International Workshop on Randomization and Computation*, pages 402–415, 2008. 3

[12] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proc. 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 474–483, 2002. 2, 3, 9

[13] O. Goldreich, editor. *Property Testing: Current Research and Surveys*, volume 6390 of *LNCS*. Springer, 2010. 1

[14] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. 1

[15] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1-2):185–204, 1998. 2, 8

[16] P. Jeavons, D. Cohen, and M. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998. 18

[17] M. Jha and S. Raskhodnikova. Testing and reconstruction of lipschitz functions with applications to data privacy. In *Proc. 52nd Annual IEEE Symposium on Foundations of Computer Science*, pages 433–442, 2011. 3

[18] I. Newman. Property testing of massively parametrized problems - a survey. In *Property testing*, volume 6390 of *LNCS*, pages 142–157. Springer, 2010. 3

[19] R. Poschel and L. A. Kaluznin. *Funktionen- und Relationenalgebren.* WEB Deutscher Verlag der Wissenschaften, Berlin, 1979. 8

[20] E. Post. *The two-valued iterative systems of mathematical logic.* Number 5 in Annals of Mathematics Studies. Princeton Univ Pr, 1941. 2, 8

[21] D. Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5:73–205, 2010. 1

[22] R. Rubinfeld and A. Shapira. Sublinear time algorithms. *Electronic Colloquium on Computational Complexity (ECCC)*, 18, 2011. TR11-013. 1

[23] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. on Comput.*, 25(2):252–271, 1996. 1

[24] T. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on the Theory of Computing*, pages 216–226, 1978. 2

[25] Y. Yoshida. Testing list H-homomorphisms. In *Proc. 27th Annual IEEE Conference on Computational Complexity*, 2012. to appear. 2, 3, 8, 9, 11, 12

A Testing Horn SAT on Trees

Let $G = (V, E)$ be a directed rooted tree. From the tree G, we can obtain an instance $\mathcal{I}$ of Horn k-SAT as follows. For every vertex $v \in V$ be a vertex and $u_1, \ldots, u_k$ its children, $\mathcal{I}$ contains a constraint of the form $(u_1 \wedge \cdots \wedge u_k \to v)$. In this section, we show that any such instance $\mathcal{I}$ of Horn k-SAT is testable with $\tilde{O}(\sqrt{n})$ queries with respect to the uniform weight function ($\mathbf{w}(i) = \frac{1}{n}$ for all $i \in [n]$).

We introduce several definitions used in this section. Given $\mathcal{I} = (V, \mathcal{C})$ an instance of Horn k-SAT on a tree, let f be an assignment for $\mathcal{I}$. A constraint $(u_1 \wedge \cdots \wedge u_k \to v) \in \mathcal{C}$ is called *violating* if $f(u_i) = 1, \forall i \in [k]$ whereas $f(v) = 0$. Let $C = (u_1 \wedge \cdots \wedge u_k \to v)$ be a violating constraint. We define the *upper violation* $u(C)$ of C as the set of ancestors w of v (including v) such that the unique path between v and w contains vertices of value 0 only. We define the *lower violation* $\ell(C)$ of C as the set of descendants w of v such that the unique path between v and w contains vertices of value 1 only. In particular, $u(C)$ contains v, and $\ell(C)$ contains $u_1, \ldots, u_k$. Also, note that $u(C)$ forms a path, and $\ell(C)$ forms a set of trees whose roots are $u_1, \ldots, u_k$. We define the *violation* $v(C)$ of C as $u(C) \cup \ell(C)$. Note that for two violating constraints C, C', their violations intersect only at their upper violations, that is, $v(C) \cap v(C') = u(C) \cap u(C')$. A vertex not contained in any violation is called *free*.

Two violating constraints C and C' are said to be *touching* if some vertex in the upper violation of C is adjacent to some vertex in the lower violation of C'. We define the *violation closure* of C is the set of vertices obtained as follows: starting with the vertices in $v(C)$, we add the vertices in violations touching the current set of violations as long as possible.

We have the following useful fact.

Lemma A.1. *Let S be a violation closure and $u, v \in S$ be vertices with $f(u) = 1$, $f(v) = 0$, and v an ancestor of u in the tree. If we perform binary search along the unique path between u and v, then we always find two vertices u', v' in a violating constraint.*

Proof. After performing binary search, we will find two vertices u' and v' such that u' is a child of v', $f(u') = 1$ and $f(v') = 0$. The only place such u' and v' exist in a violation closure is a violating constraint. □

A pair of vertices (u, v) is called *good* if u and v are in the same violation closure, $f(u) = 1, f(v) = 0$ and v is an ancestor of u. The main technical lemma here is the following.

Lemma A.2. *If f is ε-far from satisfying $\mathcal{I}$, then there exists a set M of good pairs of vertices of size εn.*

We first see that we can construct a testing algorithm using Lemma A.2.

Theorem A.3. *Horn k-SAT on trees is testable with query $O(\sqrt{n/\varepsilon} \cdot (k + \log n))$ queries.*

Proof. Let $\mathcal{I} = (V, \mathcal{C})$ be the Horn k-SAT instance on a tree, and let f be the given assignment. The algorithm is as follows.

```
1:  Let $S$ be a set of $\Theta(\sqrt{n/\varepsilon})$ vertices chosen uniformly at random.
2:  Let $S_0 = \{v \in S \mid f(v) = 0\}$ and $S_1 = \{v \in S \mid f(v) = 1\}$.
3:  for each $u \in S_1$ do
4:     Let $v$ be the closest ancestor of $u$ in $S_0$ if such a vertex exists.
5:     if $v$ exists then
6:        Perform binary search along the path between $u$ and $v$.
7:        Let $u'$ and $v'$ be the obtained vertices.
8:        if the constraint containing $u'$ and $v'$ is violating then
9:           Reject.
10: Accept.
```

It is easy to observe that the algorithm is a one-sided error tester and its query complexity is $O(\sqrt{n/\varepsilon} \cdot (k + \log n))$.

Suppose that f is ε-far from satisfying $\mathcal{I}$. Then, we have a set of εn good pairs from Lemma A.2. Thus, by choosing the hidden constant in $\Theta(\sqrt{n/\varepsilon})$ large enough, we have a good pair (u, v) in $S \times S$ with probability at least 2/3. Note that if (u, v) is a good pair and there is a vertex v' with $f(v') = 0$ along the path between u and v, then (u, v') is also a good pair. Thus, the algorithm finds a violating constraint from Lemma A.1. □

A.1 Proof of Lemma A.2

To show that there is a large set of good pairs, we consider fixing the current assignment to a satisfying assignment. An issue here is that two violation closures may overlap at their upper violations. Thus, if we tried to fix two violation closures simultaneously, then it may conflict.

To avoid this issue, we take vertices greedily. We start with an empty set $\mathcal{T}$. We choose an arbitrary violation closure S, add $S \setminus \bigcup_{T \in \mathcal{T}} T$ to $\mathcal{T}$, and repeat. We call each set in $\mathcal{T}$ a *maximal violation closure.*

Let T be a maximal violation closure. We say an assignment f' *proper* for T if $f'(u) = f'(v)$ holds for any u, v in the violation of the same violating constraint. For a set of vertices $T \subseteq V$, we define $\mathcal{I}|_T$ as the partial instance obtained by restricting the set of vertices to T. If there is a constraint $(u_1 \wedge \cdots \wedge u_k \to v)$ and $T = \{u_{i_1}, \ldots, u_{i_j}, v\}$, then the constraint becomes $(u_{i_1} \wedge \cdots \wedge u_{i_j} \to v)$ in $\mathcal{I}|_T$.

Lemma A.4. *Let f' be an assignment such that, for every maximal violation closure $T \in \mathcal{T}$, f' is proper for T and $\mathcal{I}|_T$ is satisfied by f'. Then, f' is a satisfying assignment.*

Proof. Suppose that there is a violating constraint $(u_1 \wedge \cdots \wedge u_k \to v)$ with respect to f'. Note that $f'(u_1) = \cdots = f'(u_k) = 1$ and $f'(v) = 0$. In particular, since f' is proper for any maximal violation closure, $\{u_i\}$ and v must be contained in different maximal violation closures (or one of them is free.) We have three cases to consider.

– Suppose v is in the upper violation of some violating constraint C. Then, $u_1, \ldots, u_k$ must be in the violation of C. This contradicts the fact that f' is proper for the maximal violation closure corresponding to C.

– Suppose v is in the lower violation of some violating constraint C. Let T be the maximal violation closure corresponding to C. If some $u_i, i \in [k]$ is contained in T, then it contradicts the fact that β' is a satisfying assignment for $\mathcal{I}|_T$.

  Thus, each u_i is either free or contained in another maximal violation closure. Suppose that there are maximal violation closures containing vertices in $\{u_i\}$. Then, since these maximal violation closures and T are touching, T must have been extended, contradiction.

  Thus, all $u_1, \ldots, u_k$ are free and it follows that $f(u_1) = \cdots = f(u_k) = 1$. Note that $f(v) = 1$ since y is in a lower violation. Then, the lower violation must have contained $u_1, \ldots, u_k$, contradiction.

– Suppose v is free. Note that $f(v) = 0$ since v is free. Also, there exists $i \in [k]$ such that $f(u_i) = 0$ and u_i is contained in a upper violation. Then, the upper violation must have contained v, contradiction.

□

Let $\mathrm{dist}(f, f')$ be the Hamming distance between f and f', and $\mathrm{dist}_{\mathrm{sat}}(f)$ be the distance from satisfying assignments. Also, let $\mathrm{dist}_{\mathrm{prop}}(f)$ be the distance from proper satisfying assignments. For a set of vertices T, we define $f|_T$ as the assignment obtained by restricting the domain to T. From Lemma A.4, we have the following.

Corollary A.5. *It holds that* $\mathrm{dist}_{\mathrm{sat}}(f) \leqslant \sum_{T \in \mathcal{T}} \mathrm{dist}_{\mathrm{prop}}(f|_T)$. □

Thus, to show that there is a large set of good pairs in an ε-far assignment, it suffices to show that there is a large set of good pairs in a maximal violation closure that is far from proper satisfying assignments.

From now on, we concentrate on a fixed maximal violation closure T. Note that a maximal violation closure T can be seen as a tree, in which each vertex corresponds to a violating constraint. Let C be a violating constraint in T. Let f^0_C (resp., $f_{C,1}$) be an assignment over $v(C)$ with $f^0_C \equiv 1$ (resp., $f^1_C \equiv 1$). Then, we define $d^0_C = d(f|_C, f^0_C)$ and $d^1_C = d(f|_C, f^1_C)$. Note that d^0_C (resp., d^1_C) is simply the number of ones (resp., zeros) in $v(C)$.

For a violating constraint C, we define $T_C \subseteq T$ as the subtree of T consisting of $v(C)$ and its descendants. Let $f^0_{C\downarrow}$ (resp., $f^1_{C\downarrow}$) be the proper satisfying assignment for T_C closest to f such that $\forall v \in v(C), f^0_{C\downarrow}(v) = 0$ (resp., $\forall v \in C, f^1_{C\downarrow}(v) = 1$). Then, we define $d^0_{C\downarrow} = d(f|_{T_C}, f^0_{C\downarrow})$ (resp., $d^1_{C\downarrow} = d(f|_{T_C}, f^1_{C\downarrow})$) Note that $d^0_{C\downarrow} = d^0_C$ and $d^1_{C\downarrow} = d^1_C$ when C is a leaf in the maximal violation closure.

We recursively make pairings from leaves of the maximal violation closure T. For each violating constraint C, first we make pairings between $u(C)$ and $\ell(C)$. The cardinality is clearly $\min\{|u(C)|, |\ell(C)|\}$. Suppose that $|u(C)| > |\ell(C)|$, that is, we have extra zeros in C. Then, if the subtree T_C contains unpaired vertices v with $f(v) = 1$, we make pairings between them and unmatched vertices in $u(C)$.

Lemma A.6. *Let T be a maximal violation closure. we have* $\max\{d^0_{C^\downarrow} - d^1_{C^\downarrow}, 0\}$ *unpaired ones in* T_C*.*

Proof. We use the induction on the number of violating constraints in T.

Suppose that T contains only one violating constraint C. Then, we have $\max\{d^0_C - d^1_C, 0\}$ unpaired ones in $T(C)$.

Suppose that the lemma holds for $1, \ldots, t-1$. Let T be a maximal violation closure with t violating constraints and C be the violating constraint corresponding to the root. Let $v_1, \ldots, v_\ell$ be leaves of the lower violation $\ell(C)$. Let $D_{i,1}, \ldots, D_{i,c_i}$ be violating constraints touching v_i. Note that $D_{i,1}, \ldots, D_{i,c_i}$ are children of C in the tree corresponding to T.

When we assign zeros to $v(C)$, at least one of $D_{i,1}, \ldots, D_{i,c_i}$ must be assigned zeros for each i. Thus, we have

$$d^1_{C^\downarrow} = d^1_C + \sum_i \min_{u_i \in \{0,1\}^{c_i}} \{d^{u_{i,j}}_{D^\downarrow_{i,j}}\}, \tag{1}$$

$$d^0_{C^\downarrow} = d^0_C + \sum_i \min_{u_i \in \{0,1\}^{c_i} \setminus (1,\ldots,1)} \{d^{u_{i,j}}_{D^\downarrow_{i,j}}\}. \tag{2}$$

Here, $u_{i,j}$ stands for the choice of values for $D_{i,j}$. Let $\hat{u}^1_i$ (resp., $\hat{u}^0_i$) be the one that achieves the minimum in $d^1_{C^\downarrow}$ (resp., $d^0_{C^\downarrow}$). We define $S_i = \{j \in [c_i] \mid \hat{u}^1_{i,j} \neq \hat{u}^0_{i,j}\}$. Note that if $\hat{u}^1_i \neq \hat{u}^0_i$, then we must have $\hat{u}^1_i = (1, \ldots, 1)$. Thus, for every $j \in S_i$, we have $\hat{u}^1_{i,j} = 1, \hat{u}^0_{i,j} = 0$ and $d^0_{D^\downarrow_{i,j}} \geqslant d^1_{D^\downarrow_{i,j}}$. Then,

$$d^0_{C^\downarrow} - d^1_{C^\downarrow} = d^0_C - d^1_C + \sum_{i \in [\ell]} \sum_{j \in S_i} (d^0_{D^\downarrow_{i,j}} - d^1_{D^\downarrow_{i,j}}) = d^0_C - d^1_C + \sum_{i \in [\ell]} \sum_{j \in S_i} \max\{d^0_{D^\downarrow_{i,j}} - d^1_{D^\downarrow_{i,j}}, 0\}. \tag{3}$$

From the inductive hypothesis, we have unpaired $\max\{d^0_{D^\downarrow_{i,j}} - d^1_{D^\downarrow_{i,j}}, 0\}$ ones in $T_{D_{i,j}}$. We consider two cases.

- $d^0_C - d^1_C \geqslant 0$: This means that we have more ones in C. Thus, we have $d^0_C - d^1_C$ unmatched ones in C and the claim holds.
- $d^0_C - d^1_C < 0$: This means that we have more zeros in C. It cancels out at most $d^1_C - d^0_C$ unmatched ones from $\{T_{D_{i,j}}\}_{i,j}$ and the claim holds.

□

Lemma A.7. *Let T be a maximal violation closure. Then, we can make a set of* $\mathrm{dist}_{\mathrm{prop}}(f)$ *good pairings.*

Proof. We use the same notation as previous lemma. Let C be the root violating constraint in T. Then, we have $\min\{d^0_{C^\downarrow}, d^1_{C^\downarrow}\} \geqslant d$. We use the induction on the number of violating constraints in T.

Suppose that T contains only one violating constraint C. Then, we clearly have $\mathrm{dist}_{\mathrm{prop}}(f)$ good pairs.

Suppose that the lemma holds for $1, \ldots, t-1$. Let T be a maximal violation closure with t violating constraints. We can make $\min\{d^0_C, d^1_C\}$ good pairs in C. Also, from the inductive hypothesis, we have at least $\sum_i \sum_j \min\{d^0_{D^\downarrow_{i,j}}, d^1_{D^\downarrow_{i,j}}\} = \sum_i \sum_j d^{\hat{u}^1_{i,j}}_{D^\downarrow_{i,j}}$ good pairs in $\{T_{D_{i,j}}\}_{i,j}$. We consider the following two cases.

– $\mathrm{dist}_{\mathrm{prop}}(f) = d^1_{C\downarrow}$: From (1), it suffices to show that we can make $\max\{d^1_C - d^0_C, 0\}$ matching pairs between C and $\{T_{D_{i,j}}\}_{i,j}$. It suffices to consider the case $d^1_C - d^0_C \geqslant 0$. Clearly, we have at least $d^1_C - d^0_C$ unmatched zeros in C. Note that we have (3) $\geqslant 0$. Thus,

$$d^1_C - d^0_C \leqslant \sum_i \sum_{j \in S_i} \max\{d^0_{D^\downarrow_{i,j}} - d^1_{D^\downarrow_{i,j}}, 0\}.$$

Thus, from Lemma A.6, we have at least $d^1_C - d^0_C$ unpaired ones in $\{T_{D_{i,j}}\}_{i,j}$.

– $\mathrm{dist}_{\mathrm{prop}}(f) = d^0_{C\downarrow}$: It suffices to show that we can make

$$\begin{aligned} & d^0_{C\downarrow} - \min\{d^0_C, d^1_C\} - \sum_i \sum_j d^{\hat{u}^1_{i,j}}_{D^\downarrow_{i,j}} \\ &= \max\{d^0_C - d^1_C, 0\} + \sum_i \sum_{j \in S_i} (d^0_{D^\downarrow_{i,j}} - d^1_{D^\downarrow_{i,j}}) \\ &= \max\{d^0_C - d^1_C, 0\} + \sum_i \sum_{j \in S_i} \max\{d^0_{D^\downarrow_{i,j}} - d^1_{D^\downarrow_{i,j}}, 0\}. \end{aligned} \tag{4}$$

good pairs between C and $\{T_{D_{i,j}}\}_{i,j}$. From the fact that (3) $\leqslant 0$, we have (4) $\leqslant \max\{d^0_C - d^1_C, 0\} + d^1_C - d^0_C = \max\{d^1_C - d^0_C, 0\}$. Thus, it suffices to consider the case $d^1_C - d^0_C \geqslant 0$. In this case, (4) $= \sum_i \sum_{j \in S_i} \max\{d^0_{D^\downarrow_{i,j}} - d^1_{D^\downarrow_{i,j}}, 0\} \leqslant d^1_C - d^0_C$. Clearly, we have at least $d^1_C - d^0_C$ unpaired zeros in C. Also, we have $\sum_i \sum_{j \in S_i} \max\{d^0_{D^\downarrow_{i,j}} - d^1_{D^\downarrow_{i,j}}, 0\}$ unpaired ones in $\{T_{D_{i,j}}\}_{i,j}$.

□

B Proof of Lemma 4.2

We first review properties of $(k+1)$-near-unanimity operations. If a relation R is preserved by a $(k+1)$-near-unanimity operation, then it is known that the relation can be made k-ary by projecting R into every k-sized subset [10, 16]. That is, a tuple $\mathbf{a}$ belongs to R if and only if $\mathbf{a}|_U \in R|_U$ for every subset U with $|U| = k$, where $\mathbf{a}|_U$ and $R|_U$ are projections of $\mathbf{a}$ and R on U, respectively. Thus, we can assume the input instance $\mathcal{I} = (V, \mathcal{C})$ consists of constraints of arity at most k.

Now, we can preprocess the instance $\mathcal{I}$ in polynomial time and we obtain a set of partial solutions $\mathcal{S}_U$ for each variable set $U \subseteq V$ of size k. A crucial property of $\mathcal{S}_U$ is that any partial solution $\mathbf{a} \in \mathcal{S}_u$ can be extended to a satisfying assignment for the entire instance. The preprocess is called $(k, k+1)$-Minimality and the property of $\mathcal{S}_U$ is called the Helly property (see [3, 10] for details). We call a subset of variables $U \subseteq V$ of size k *violated* with respect to an assignment $f : V \to \{0, 1\}$ if $f|_U \notin \mathcal{S}_U$.

Proof of Lemma 4.2. First, we describe our algorithm. We query each variable v with probability $q \cdot \mathbf{w}(v)$, where $q = \Theta(\frac{n^{1-1/k}}{\varepsilon})$. If we query more than $100q$ times along the way, we immediately stop and accept. Suppose that the number of queries is at most $100q$. Then, we reject if there is some subset $S \subseteq V$ of size k such that $f|_U \notin \mathcal{S}_U$, and we accept otherwise.

It is easy to see that the algorithm always accepts if f is a satisfying assignment (no matter whether we stopped as we have queried more than $100q$ times).

Now, we see that the algorithm rejects with high probability when the instance is ε-far. From Markov's inequality, the query complexity is at most $100q$ with probability at least $\frac{99}{100}$.

Let $\mathcal{U}$ be the family of violated variable sets U of size k. We first observe that $f|_{V \setminus H}$ is extendable to a satisfying assignment for any hitting set H of $\mathcal{U}$. Indeed, if $f|_{V \setminus H}$ is not extendable, then there must be a variable set $U \subseteq V \setminus H$ with $|U| = k$ and $f|_U \notin \mathcal{S}_U$ from the Helly property. However, such set U must be contained in $\mathcal{U}$, contradiction.

Let V_ℓ be a set of variables v with $\mathbf{w}(v) \leqslant \frac{\varepsilon}{2n}$ and $\mathcal{U}_h \subseteq \mathcal{U}$ be a set of violated variable sets that is not hit by any vertex in V_ℓ. Let H_h^* be a minimum (weighted) hitting set of $\mathcal{U}_h$. From the argument above, $f|_{V \setminus (V_\ell \cup H_h^*)}$ is extendable to a satisfying assignment and hence we have $\mathbf{w}(V_\ell \cup H_h^*) \geqslant \varepsilon$. Since, $\mathbf{w}(V_\ell) \leqslant \frac{\varepsilon}{2n} \cdot n = \frac{\varepsilon}{2}$, we have $\mathbf{w}(H_h^*) \geqslant \frac{\varepsilon}{2}$. Let $\mathcal{U}_h^* \subseteq \mathcal{U}_h$ be any maximal set of disjoint violated variable sets in $\mathcal{U}$. Since the variable set $\bigcup_{U \in \mathcal{U}_h^*} U$ itself is a hitting set of $\mathcal{U}_h$, we have $\sum_{U \in \mathcal{U}_h^*} \sum_{v \in U} \mathbf{w}(v) \geqslant \mathbf{w}(H_h^*) \geqslant \frac{\varepsilon}{2}$.

We need the following claims for our analysis.

Claim B.1. *Suppose $\sum_{i \in [k]} x_i = s$ and $\min_{i \in [k]} x_i \geqslant t \geqslant 0$. Then, $\prod_{i=1}^k x_i \geqslant t^{k-1}(s - (k-1)t)$.*

Proof. Use induction on k. □

Claim B.2. *Suppose $\sum_{j \in [\ell]} \sum_{i \in [k]} x_{ij} \geqslant s$ and $\min_{i \in [k], j \in [\ell]} x_{ij} \geqslant t \geqslant 0$. Then, $\sum_{j \in [\ell]} \prod_{i \in [k]} x_{ij} \geqslant t^{k-1}s - t^k(k-1)\ell$.*

Proof. For $j \in [\ell]$, let $s_j = \sum_{i \in [k]} x_{ij}$. From Claim B.1, we have

$$\sum_{j \in [\ell]} \prod_{i \in [k]} x_{ij} \geqslant \sum_{j \in [\ell]} t^{k-1}(s_j - (k-1)t) \geqslant t^{k-1}s - t^k(k-1)\ell.$$ □

Now, we are back to the proof of Lemma 4.2. Note that for each variable set U, the probability that we do not find U is $1 - q^k \prod_{v \in U} \mathbf{w}(v)$. Thus, the probability that we do not find any violated variable set in $\mathcal{U}_h$ during the process of our algorithm is

$$\begin{aligned} \prod_{U \in \mathcal{U}_h^*} (1 - q^k \prod_{v \in U} \mathbf{w}(v)) &\leqslant \prod_{U \in \mathcal{U}_h^*} \exp(-q^k \prod_{v \in U} \mathbf{w}(v)) = \exp(-q^k \sum_{U \in \mathcal{U}_h^*} \prod_{v \in U} \mathbf{w}(v)) \\ &\leqslant \exp\left(-q^k \left(\left(\frac{\varepsilon}{2n}\right)^{k-1} \varepsilon - \left(\frac{\varepsilon}{2n}\right)^k (k-1)|\mathcal{U}_h^*|\right)\right) \quad \text{(from Claim B.2)} \\ &\leqslant \exp\left(-q^k \left(\left(\frac{\varepsilon}{2n}\right)^{k-1} \varepsilon - \left(\frac{\varepsilon}{2n}\right)^k \frac{(k-1)n}{k}\right)\right) \quad \text{(from } |\mathcal{U}_h^*| \leqslant \frac{n}{k}) \\ &\leqslant \exp\left(-\frac{(\varepsilon q)^k}{2(2n)^{k-1}}\right) \end{aligned}$$

If we choose the constant hidden in q large enough, then assuming that n is sufficiently large enough, the probability above is bounded by $\frac{1}{100}$. Thus, with probability at least $\frac{98}{100}$, we reject the instance. □

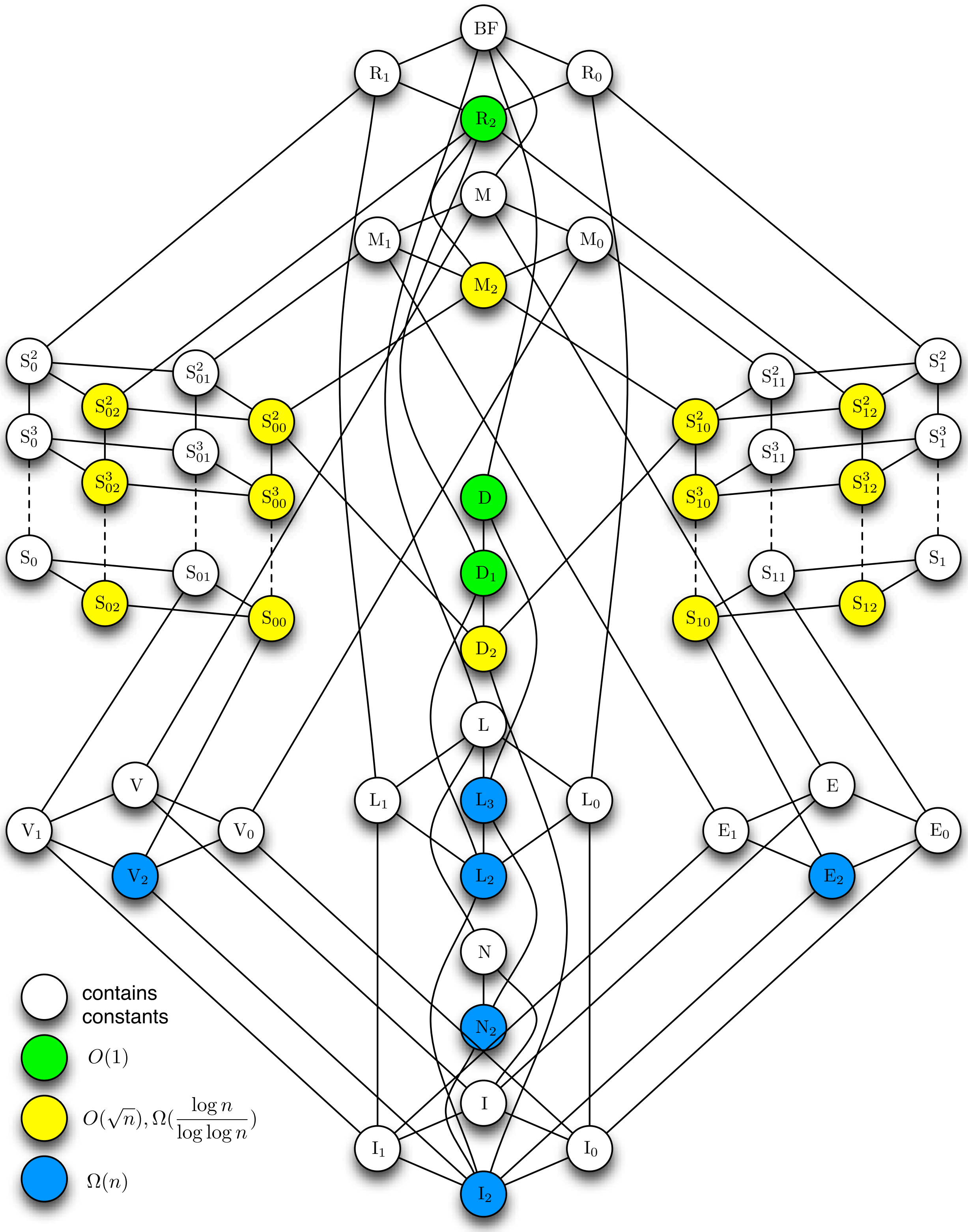

Figure 1: Post's Lattice and our result

Table 1: List of all closed classes of Boolean functions and their bases. $h_n(x_1,\ldots,x_{n+1}) = \bigvee_{i=1}^{n+1} x_1 \wedge x_2 \wedge \cdots \wedge x_{i-1} \wedge x_{i+1} \wedge \cdots \wedge x_{n+1}$ is an $(n+1)$-ary near-unanimity operation.

Name	Definition	Base
BF	All Boolean functions	$\{\vee,\wedge,\neg\}$
R_0	$\{f \in BF \mid f$ is 0-reproducing$\}$	$\{\wedge,\oplus\}$
R_1	$\{f \in BF \mid f$ is 1-reproducing$\}$	$\{\vee,\leftrightarrow\}$
R_2	$R_1 \cap R_0$	$\{\vee, x \wedge (y \leftrightarrow z)\}$
M	$\{f \in BF \mid f$ 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	$\{f \in BF \mid f$ is 0-separating of degree $n\}$	$\{\rightarrow, \mathrm{dual}(h_n)\}$
S_0	$\{f \in BF \mid f$ is 0-separating$\}$	$\{\rightarrow\}$
S_1^n	$\{f \in BF \mid f$ is 1-separating of degree $n\}$	$\{x \wedge \overline{y}, h_n\}$
S_1	$\{f \in BF \mid f$ is 1-separating$\}$	$\{x \wedge \overline{y}\}$
S_{02}^n	$S_0^n \cap R_2$	$\{x \vee (y \wedge \overline{z}), \mathrm{dual}(h_n)\}$
S_{02}	$S_0 \cap R_2$	$\{x \vee (y \wedge \overline{z})\}$
S_{01}^n	$S_0^n \cap M$	$\{\mathrm{dual}(h_n), 1\}$
S_{01}	$S_0 \cap M$	$\{x \vee (y \wedge z), 1\}$
S_{00}^n	$S_0^n \cap R_2 \cap M$	$\{x \vee (y \wedge z), \mathrm{dual}(h_n)\}$
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	$\{f \mid f$ is self-dual$\}$	$\{x\overline{y} \vee x\overline{z} \vee \overline{yz}\}$
D_1	$D \cap R_2$	$\{xy \vee x\overline{z} \vee y\overline{z}\}$
D_2	$D \cap M$	$\{xy \vee yz \vee xz\}$
L	$\{f \mid f$ 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	$\{f \mid f$ 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	$\{f \mid f$ 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	$[\{\mathrm{id}\}]$	$\{\mathrm{id}\}$

ECCC ISSN 1433-8092

http://eccc.hpi-web.de