Testing Equivalence to the Hamiltonian Cycle Polynomial

Agrim Dewan
Indian Institute of Science, Bengaluru, India

Abstract

The Hamiltonian Cycle polynomial, denoted as HC_n, is defined to be the sum of the weighted Hamiltonian Cycles in an n-vertex complete digraph, with vertices labeled 1 to n and edges weighted by formal variables $x_{i,j}$. The Permanent and HC, defined as the family $\{HC_n | \ n \geq 1\}$, were studied by Valiant (STOC 1979), with the former shown to be VNP-complete over all fields of characteristic other than 2, and the latter to be VNP-complete over *every* field. Since its introduction, HC has been studied from the perspective of circuit lower bounds by Jerrum-Snir (JACM 1982), determinantal complexity by Huttenhain-Ikenmeyer (LAA 2016), and its connection with the Permanent and the Determinant polynomials by Goulden-Jackson (EJC 1981) and Grochow (ToC 2017). It has been the most prominent choice for generalising results to all fields, such as in Malod (CCC 2007) and Grochow-Mulmuley-Qiao (ICALP 2016), owing to its VNP-completeness over every field. Hrubes (ToCT, 2016) showed the VNP-completeness of many graph-based polynomial families over every field by using HC.

In Kayal (STOC 2012), a randomised polynomial time algorithm was given for the following problem: Given an n^2-variate degree-n polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ as a black box, decide if there exists $A \in \mathrm{GL}_{n^2}(\mathbb{F})$ such that $f(\mathbf{x}) = Perm_n(A\mathbf{x})$. Here, the Permanent polynomial $Perm_n$ computes the permanent of the $n \times n$ symbolic matrix $(x_{i,j})$. This problem is known as testing equivalence to the Permanent, or alternatively, ET for Permanent.

In this work, we study ET for HC. While both families are VNP-complete, the efficient ET algorithm for Permanent does not imply the same for HC. Besides, there are crucial differences between the two polynomials that make studying the complexity of ET for HC interesting: The underlying decision problem corresponding to the Permanent is in P (detecting perfect matchings in a bipartite graph), but that for HC (detecting Hamiltonian cycles in a digraph) is NP-complete. The Permanent polynomial is known to be characterised by its symmetries as shown by Mulmuley-Sohoni (SIAM J. Computing, 2001). This property yields an *efficient* algorithm for the circuit-testing problem for the Permanent, a special case of ET for the Permanent, in which we check whether a given circuit computes the Permanent. In contrast, we show HC_n is *not* characterised by its symmetries.

In this work, we give a randomised polynomial time ET algorithm for HC with mild constraints on the underlying field. The algorithm is obtained by studying and completely characterising the Lie algebra and the symmetries of HC_n. We show that, like the Permanent polynomial, the symmetries of HC_n are generated by permutation and scaling matrices over large enough fields. However, we also show that, unlike the Permanent polynomial, HC_n is *not* characterised by its symmetries. Nevertheless, like the Permanent polynomial, HC_n is downward self-reducible, as shown in Zhang-Bai (TCS 2011), which implies HC_n is characterised by circuit identities and that we can efficiently test whether a given circuit C computes HC_n. We also get a Flip theorem for HC_n as a result of its circuit identities.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Equivalence Testing, Hamiltonian Cycle Polynomial, Symmetries, Lie Algebra, Circuit identities

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.90

Related Version *Full Version*: `https://arxiv.org/abs/2606.26653` [11]

51st International Symposium on Mathematical Foundations of Computer Science (MFCS 2026).
Editors: Michal Koucký and Daniela Petrişan; Article No. 90; pp. 90:1–90:17

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements The author thanks Chandan Saha for suggesting the problem and Abhiram Aravind for going through the proofs, which helped improve the presentation of the paper. The author also thanks the anonymous reviewers of MFCS 2026 for their constructive and informative feedback, particularly one reviewer for pointing to the Pascal Determinant polynomial family, which is VNP-complete and for which an ET algorithm can be designed using knowledge of its continuous symmetries.

1 Introduction

Two n-variate polynomials $f, g \in \mathbb{F}[\mathbf{y}]$ are *equivalent*, denoted by $f \sim g$, if $f(\mathbf{y}) = g(A\mathbf{y}+\mathbf{b})$ for some invertible $A \in \mathrm{GL}_n(\mathbb{F})$ and $\mathbf{b} \in \mathbb{F}^n$. Clearly, the relation $f \sim g$ is an equivalence relation. The Polynomial Equivalence problem (PE) involves deciding for two given polynomials f and g, if $f \sim g$ holds or not. Here, the inputs are assumed to be given as lists of coefficients. On one hand, PE is known to be at least as hard as Graph Isomorphism [1, 31]. On the other hand, the known upper bound on its complexity depends on the underlying field. Over finite fields, PE is in NP $\cap$ co-AM and hence unlikely to be NP-complete, while over $\mathbb{Q}$ we do not even know if it is decidable [45, 47]. Even when both inputs are restricted, limited results are known, see [11] for a detailed discussion. Thus, understanding the complexity of PE remains a challenge. This has led to a natural variant of PE, called *equivalence testing*, being studied in the literature to make progress towards addressing this challenge.

Equivalence testing. PE asks to check the equivalence of two arbitrary polynomials. In contrast, for Equivalence Testing (ET), we first fix a polynomial family $\mathcal{F}$ or a circuit[1] class $\mathcal{C}$, the goal then is to decide for a given *single* polynomial f, whether $f \sim g$, for some g in $\mathcal{F}$ or in $\mathcal{C}$, respectively. The two versions are called ET for polynomial families and ET for circuit classes, respectively. The study of ET for polynomial families was initiated in [31, 32], where efficient ET algorithms were given for the Power Symmetric and Elementary Symmetric polynomials, the Permanent and the Determinant. Since then, efficient ET algorithms have been given for various important polynomial families. The ET algorithms are efficient even if f is given as a black-box[2] or as a circuit. The study of ET for circuit classes is more recent, with the authors of [19] showing an efficient ET for read-once formulas (ROFs). The authors of [3] showed that ET for sparse polynomials (depth-2 circuits) is NP-hard. Later, ET for read-once oblivious algebraic branching programs (ROABPs) was also shown to be NP-hard [44, 5]. See [11] for more results on ET for polynomial families and circuit classes.

Among all the polynomial families for which ET has been studied so far, the Permanent (see Definition 1) is the most prominent VNP-complete family.[3] An efficient ET algorithm for a VNP-complete family does not imply the same for other VNP-complete families. Since ET has been shown to be NP-hard for some natural circuit classes, a natural question arises:

Is ET "hard" for some "natural" VNP-complete polynomial family?

[1] By a circuit we mean an *arithmetic circuit*, unless stated otherwise. An arithmetic circuit is like a Boolean circuit except it uses $+$ or $\times$ gates in place of AND, OR, and NOT gates, and the edges are labeled by $\mathbb{F}$-elements. The circuit computes a polynomial over $\mathbb{F}$.

[2] Black-box access to a polynomial f means we are given oracle access to f. Thus, we can query f at any point $\mathbf{a}$ of our choosing and obtain $f(\mathbf{a})$ in unit time.

[3] The classes VP and VNP are the algebraic analogues of P and NP, respectively, see [48, 8] for a rigorous definition. The authors of [18] studied ET for the Nisan-Wigderson polynomial (NW), which is in VNP but not known to be VNP-complete. The Pascal determinant, see [20] and Chapter 8 in [35], is a generalisation of the Determinant and is VNP-complete. The Pascal Determinant is characterised by its continuous symmetries [35], due to which an ET algorithm for the Pascal Determinant follows almost similarly to that for the Determinant given in Corollary 4.6.4 of [15].

The Hamiltonian Cycle polynomial family, defined below, is another natural VNP-complete family.

▶ **Definition 1** (Permanent and Hamiltonian Cycle polynomial families). *Let $n \geq 1$. The Permanent polynomial $Perm_n(X_n)$ and the Hamiltonian Cycle polynomial $HC_n(X_n)$ are homogeneous degree-n polynomials defined on the $n \times n$ matrix of variables $X_n = (x_{i,j})_{i,j\in[n]}$ as:*

$$Perm_n(X_n) := \sum_{\sigma\in S_n} \prod_{i\in[n]} x_{i,\sigma(i)} \quad \text{and} \quad HC_n(X_n) := \sum_{\sigma\in C_n} \prod_{i\in[n]} x_{i,\sigma(i)}, \tag{1}$$

respectively, with HC_1 defined as the zero polynomial. Here, S_n and $C_n \subsetneq S_n$ are the set of all permutations on $\{1, 2 \dots n\}$, and the set of all cyclic *permutations on $\{1, 2 \dots n\}$, respectively. We denote by Perm the family $\{Perm_n |\ n \geq 1\}$ and by HC the family $\{HC_n |\ n \geq 1\}$.*

Treating X_n as the adjacency matrix of a complete digraph G with edge weights as formal variables, $HC_n(X_n)$ is the sum of the weights of all the Hamiltonian cycles of G. The Permanent polynomial is the sum of all the weights of perfect matchings in a bipartite graph, whose biadjacency matrix is X_n. Observe that though HC_n is defined on n^2 many variables, it depends only on the off-diagonal $n^2 - n$ entries, which we denote as $\mathbf{x}$. We treat HC_n as an $(n^2 - n)$-variate polynomial in $\mathbf{x}$ variables, and the evaluation of HC_n at an $n \times n$ matrix A is denoted by $HC_n(A)$, where the diagonal entries of A will be ignored.

After the Permanent, HC seems to be the most prominent example of a VNP-complete family. In [48], HC was shown to be VNP-complete over *every* field, while the VNP-completeness of Permanent holds only over fields of characteristic other than 2. The proof of VNP-completeness of HC by [48] does *not* rely on the VNP-completeness of the Permanent. In fact, HC is VNP-complete even over rings [38]. There have been various works where HC has been studied in multiple contexts, such as lower bounds [27, 28, 2, 23], and its connection to the Permanent and the Determinant [14, 8, 16]. It has been the main choice as a VNP-complete family over all fields whenever such a family was sought to generalize results to all fields [39, 34, 17, 24, 25, 12]. The author of [22] used the completeness of HC over every field to show the existence of more VNP-complete families over every field. See [11] for a detailed discussion on works involving HC. Thus, HC is a well-studied, natural, and fundamental VNP-complete family.

In this work, we study the Equivalence Testing problem for HC, defined as follows:

▶ **Problem 2** (ET for HC). *Given black box access to an $n^2 - n$-variate degree-n polynomial $f(\mathbf{x})$, decide if there exists $A \in \mathrm{GL}_{n^2-n}(\mathbb{F})$ and $\mathbf{b} \in \mathbb{F}^{n^2-n}$ such that $f(\mathbf{x}) = HC_n(A\mathbf{x} + \mathbf{b})$ and return A and $\mathbf{b}$ if they exist.*

ET for HC_n. While both the Permanent and HC are VNP-complete, the efficient ET algorithm for Permanent does not imply the same for HC.[4] It is a priori unclear why ET for HC could be "easy" because there are crucial differences between the Permanent and HC. First, the complexity of the *zero-testing problem* for the Permanent and HC is vastly different. The zero-testing problem for a polynomial family $\mathcal{F} = \{f_n\}$, where f_n is n-variate and has integer coefficients, is the task of checking for a *given* point $\mathbf{a} \in \{0, 1\}^n$, whether

[4] The completeness of the Permanent and HC is with respect to p-projections [48], which means any polynomial $f(\mathbf{y})$ can be obtained from $Perm_m(X)$ (or HC_n) by replacing each variable in $Perm_m$ by a $\mathbf{y}$ variable or a field constant. In contrast, $f \sim Perm_m$ happens via *invertible* linear transforms on the variables. Thus, $g \sim HC_n$ does *not* necessarily mean that $g \sim Perm_m$ for some m.

$f(\mathbf{a})$ is 0 or not over $\mathbb{Z}$. Note that f is *not* given to us in any form, that is, we don't even have black-box access to f. Zero-testing for the Permanent lies in P because it is the task of detecting a perfect matching in a bipartite graph.[5] Zero-testing for HC is NP-complete because it amounts to detecting a Hamiltonian Cycle in a digraph.[6] In both cases, the graph is given as a 0/1 matrix. Second, the Permanent polynomial is known to be characterised by its symmetries [43], see Definition 21, and [15] for a proof, but we show HC_n is *not* characterised by its symmetries in Theorem 8. This becomes important in the context of the *circuit testing problem* for a polynomial family $\mathcal{F}$ which asks to check if a given circuit C computes f_n for some n, i.e., is $\mathrm{C}(\mathbf{y}) = f(\mathbf{y})$? The circuit testing problem for a polynomial family $\mathcal{F}$ is a special case of ET for $\mathcal{F}$, with A as the identity matrix and $\mathbf{b} = \mathbf{0}$. Efficient circuit testing algorithms for the Permanent follow by the downward self-reducibility[7] of $Perm_n$ [29] and also due to characterisation by symmetries of $Perm_n$ [43], see Page 81 in [15] for another proof. Since HC_n is not characterised by its symmetries, one cannot hope to perform circuit testing efficiently for it via symmetries. Given these crucial differences between the two polynomials, it is natural to ask:

Is ET "easy" for the Hamiltonian Cycle polynomial?

In this work, we answer this question positively by showing a randomised polynomial time ET algorithm for HC (see Theorem 3). To our knowledge, an efficient ET for HC was *not* known before this work.

The author of [32] developed the ET algorithm for the Permanent by using the knowledge of the symmetries and the Lie algebra of the Permanent, which have been well studied [40, 7]. We follow the approach of [32] and study the Lie algebra and symmetries of HC_n to design an ET algorithm for it, though there are challenges to overcome in analysing the Lie algebra and designing the ET algorithm. The Permanent has some "nice" monomials which are leveraged by [32] in his algorithm, but these monomials are absent in HC_n. Thus, we deviate significantly from the implementation of the scaling matrix recovery step in [32] and instead use ideas from [18]. The last step of the ET algorithm for $Perm_n$ uses circuit testing for a soundness check. Since HC_n is not characterised by its symmetries, we cannot hope to perform circuit testing for it via this route. To perform circuit testing for HC, we show it is characterised by circuit identities (Theorems 10 and 11) via its downward self-reducibility (Lemma 4 in [49]). See Section 1.2 for a discussion on the proof ideas, and [11] for a detailed comparison with [32, 18]. Table 1 compares the results with those of the Permanent.

1.1 Our Results

We first state our assumptions on the computational model. Over $\mathbb{Q}$ and finite fields $\mathbb{F}_q$, we assume the Turing machine model. Over fields like $\mathbb{R}$ and $\mathbb{C}$, we assume a model similar to the BSS model [6], where one can perform an arithmetic operation in unit time. We also assume access to an efficient black-box univariate polynomial factorisation algorithm, an

[5] In fact, it is in quasi-NC^2 [13] and very recently it has been shown to be in NC [9].

[6] Similarly, the Permanent has an FPRAS [26] while no FPRAS exists for HC unless NP=RP. An FPRAS for the Permanent is a randomised algorithm which takes as input a non-negative $n \times n$ matrix A and a parameter $0 < \epsilon < 1$, and with high probability outputs a value in the range $[(1-\epsilon)Perm_n(A), (1+\epsilon)Perm_n(A)]$. The running time is $(n\frac{1}{\epsilon})^{O(1)}$.

[7] For the Permanent, downward self-reducibility means that computing the Permanent of $n \times n$ matrix polytime Turing reduces to computing the Permanent of n many $(n-1) \times (n-1)$ matrices, which follows by the cofactor expansion of the Permanent. In general, a problem is said to be downward self-reducible if solving an instance I of size n Turing reduces to solving instances of smaller size.

Table 1 A comparison between Permanent and HC_n.

Property	HC_n ***(This work)***	***Permanent***
Efficient ET algorithm?	Yes	Yes
Explicit basis of Lie algebra known?	Yes	Yes
Characterisation by symmetries?	**No**, for $n \geq 5$	**Almost all fields**
Symmetries generated by PS matrices?	Yes	Yes
Characterisation by circuit identities?	Yes	Yes
Scaling symmetries continuous?	Almost all fields, except for $n = 4$ over $\mathbb{Q}, \mathbb{R}, \mathbb{C}$	Almost all fields
Flip Theorem known?	Yes	Yes
Complexity of Zero testing?	**NP-complete**	**P**

assumption which is justified over fields $\mathbb{F}_q$ [4] and $\mathbb{Q}$ [36]. We consider HC_n for $n \geq 3$. For $n = 2$, HC_n is a monomial, and an efficient ET algorithm follows for it by the factorisation algorithm of [30].

▶ **Theorem 3** (ET for HC). *Let $\mathbb{F}$ be a field where $|\mathbb{F}| > 3n^5$ and $char(\mathbb{F}) = 0$ or $char(\mathbb{F}) > n$. There is a randomised $(n\beta)^{O(1)}$ time algorithm which, when given black-box access to an $(n^2 - n)$-variate, homogeneous degree-n polynomial $f(\mathbf{x})$ with β as the bit complexity of the coefficients, decides correctly, with high probability, if there exists $A \in \mathrm{GL}_{n^2-n}(\mathbb{F})$ such that $f(\mathbf{x}) = HC_n(A\mathbf{x})$ and outputs A if it exists. Over finite fields, the running time is randomised $(n \log(|\mathbb{F}|))^{O(1)}$.*

▶ Remark 4. The algorithm also works for the more general case where $f = HC_n(A\mathbf{y} + \mathbf{b})$, $A \in \mathbb{F}^{(n^2-n)\times m}$ is a full row rank matrix (hence $n^2 - n \leq m$) and $\mathbf{b} \in \mathbb{F}^{n^2-n}$ is a translation vector. See Theorem 28 in [32] and Appendix B in [33], which show how to appropriately modify the ET algorithm to handle the general case.

▶ Remark 5. The constraint on the characteristic arises from computing second-order derivatives of f and is imposed to ensure that these derivatives don't vanish. The constraint on the field size is due to the use of the Polynomial Identity Lemma [10, 37, 46, 50].

The ET algorithm uses the knowledge of the group of symmetries $\mathcal{G}_{HC_n}$ and the Lie Algebra $\mathfrak{g}_{HC_n}$ of HC_n, see Definitions 15 and 16. In Propositions 22 and 25, we completely characterise and give a basis for $\mathfrak{g}_{HC_n}$, while Theorem 6 gives the general structure of $\mathcal{G}_{HC_n}$.

▶ **Theorem 6** (Structure of $\mathcal{G}_{HC_n}$). *Let $|\mathbb{F}| > \binom{n^2-n}{2}$. If A is a symmetry of HC_n, then $A = PS$ for some permutation matrix P and scaling matrix S.*

▶ Remark 7. Propositions 30 and 32 describe the permutation and scaling symmetries of HC_n respectively. If we treat HC_n as a polynomial in all the variables of X_n, then any $A \in \mathcal{G}_{HC_n}$ acts on X_n as $A(X_n) = PLX_nRP^T$ or $A(X_n) = PLX_n^TRP^T$, where P is a permutation matrix, and L and R are diagonal matrices such that $Perm_n(RL) = 1$. In contrast, any $A \in \mathcal{G}_{Perm_n}$ acts as $A(X_n) = PLX_nRQ$ or $A(X_n) = PLX_n^TRQ$, where P and Q are permutation matrices and L, R are as before.

Unlike $Perm_n$, HC_n is *not* characterised by its symmetries, see Definition 21.

▶ **Theorem 8** (Non-characterisation by symmetries). *Let $n \geq 5$ and $\mathbb{F}$ be such that $|\mathbb{F}| > \binom{n^2-n}{2}$. Then, HC_n is* not *characterised by its symmetries over $\mathbb{F}$.*

▶ Remark 9. In [11], we show that HC_n, for $n = 3$ and 4, is characterised by its symmetries over appropriate fields $\mathbb{F}$.

However, it turns out that HC_n is characterised by circuit identities, see Definition 20, over *any* field $\mathbb{F}$. This follows from the downward self-reducibility of HC_n as shown in Lemma 35, and Lemma 4 of [49].

▶ **Theorem 10** (Circuit Identities). *Over any field $\mathbb{F}$, HC_n is characterised by circuit identities.*

As a consequence of Theorem 10, we get Theorems 11 and 12. We use Theorem 11 to prove the soundness of the ET algorithm given by Theorem 3.

▶ **Theorem 11** (Circuit Testability). *Let C be a circuit in $n^2 - n$ variables, of size s, and degree of output polynomial $d = s^{O(1)}$ over a field $\mathbb{F}$, where $|\mathbb{F}| > 2d$. There is a randomised $(ns)^{O(1)}$ time algorithm over $\mathbb{F}$ which, when given black box access to C, decides correctly with high probability whether $\mathrm{C}(\mathbf{x}) = HC_n(\mathbf{x})$ or not.*

A flip theorem essentially states that if some function is not computable by small-size circuits, then we can efficiently construct a short list of counterexamples such that any small-size circuit fails on some counterexample in the list. Flip theorem is known for the Permanent polynomial [41, 42] and the Nisan-Wigderson polynomial [18].

▶ **Theorem 12** (Flip Theorem for HC). *Let $|\mathbb{F}| > n^{O(1)}$. Suppose HC_n is not computable by $n^{O(1)}$ size circuits over $\mathbb{F}$. There is a randomised $n^{O(1)}$ time algorithm which on input 1^{n^2-n} outputs $n-1$ matrices $A_1, \ldots A_{n-1}$, where $A_i \in \mathbb{F}^{n \times n}$, such that, with high probability, for any polynomial size circuit C there is an $i \in [n-1]$, $\mathrm{C}(A_i) \neq HC_n(A_i)$. Further, derandomisation of black-box polynomial identity testing implies that the A_i's can be computed in deterministic $n^{O(1)}$ time.*

▶ Remark 13. Note that Theorem 11 is *not* implied by Theorem 12 because HC_n is not known to be computable by $n^{O(1)}$ size circuit. Verifying $\mathrm{C}(A_i) \neq HC_n(A_i)$ for some i needs the evaluations $HC_n(A_i)$, which can not be done efficiently.

1.2 Proof Techniques

We follow the Lie algebraic approach of [32], which was used to obtain an ET algorithm for the Permanent. We describe the ET algorithm for the HC at a high level. Suppose we are given black-box access to an $n^2 - n$-variate degree-n polynomial $f(\mathbf{x})$. The algorithm has four steps:

1. *Reduction to PS-equivalence testing.* Using the structure of the Lie algebra of HC_n, compute an invertible D such that $f(D\mathbf{x}) = HC_n(PS\mathbf{x})$ for some permutation P and scaling S, assuming $f \sim HC_n$. See Algorithm 2.
2. *Reduction to S-equivalence testing.* Let $f_1(\mathbf{x}) = f(D\mathbf{x})$. Using the knowledge of the permutation symmetries of HC_n, and the set of vanishing second-order partial derivatives of HC_n, recover P so that $f_1(P\mathbf{x}) = HC_n(S\mathbf{x})$, assuming $f \sim HC_n$. Thus, $f_1(P\mathbf{x})$ is S-equivalent to HC_n. See Algorithm 3.
3. *Recovering S.* Let $f_2(\mathbf{x}) = f_1(P\mathbf{x})$, assuming $f \sim HC_n$. Query f_2 at $k = O(n^2)$ many points to get constants $c_1, c_2, \ldots, c_k$, and solve a system of linear equations where the variables are the entries of S which we wish to recover. See Algorithm 4.
4. *Verification \ Soundness Test.* Let $B = DPS$. Use the downward self-reducibility, Lemma 35 or Lemma 4 of [49], of HC_n to verify if $f(B\mathbf{x}) = HC_n(\mathbf{x})$. If so, output B^{-1}. See Algorithm 1.

Though the ET algorithm for HC is obtained via the Lie algebraic approach, like it was done for the Permanent in [32], there are two major differences between the algorithms. First, the Lie algebra and the symmetries of the Permanent have been well-studied [7, 40], and are

leveraged by [32]. To our knowledge, the Lie algebra and symmetries of HC_n have not been studied before. Therefore, we first study and give a complete characterisation of the Lie algebra $\mathfrak{g}_{HC_n}$ of HC_n over all fields (see Propositions 22 and 25) by using ideas from [18] to construct a basis for $\mathfrak{g}_{HC_n}$; we also characterise the group of symmetries $\mathcal{G}_{HC_n}$ over large enough fields (Proposition 29). Second, we deviate from [32] in the execution of Step 3. The deviation occurs because of an important difference in the monomials of HC_n and those of $Perm_n$. In $Perm_n$, there are enough pairs of monomials which differ in exactly 2 variables and are leveraged by [32] to recover S by querying the scaled Permanent polynomial at these monomials. In contrast, *any* two monomials of HC_n differ in at least 3 variables (see Claim 24), which makes it unclear if they can be leveraged in the same way as for $Perm_n$ to recover S. Instead, we follow the approach in [18] to recover S. There are other key technical differences between our analysis and that in [32, 18], see [11] for a detailed comparison. Refer to Table 1 for a comparison of results between HC_n and the Permanent.

To prove Theorem 6, we leverage the structure of $\mathfrak{g}_{HC_n}$ and conjugacy of Lie algebras of equivalent polynomials (Lemma 17). We prove Theorem 8 by using the knowledge of the permutation (Proposition 32) and scaling symmetries (Proposition 30) of HC_n. Theorem 10 is proved by using the downward self-reducibility of HC_n and an adaptation of Lemma 7.13 from [8]. Theorems 11 and 12 then follow by using the identities established in Theorem 10.

2 Preliminaries

2.1 Notations and Definitions

The set of natural numbers is $\mathbb{N} = \{1, 2 \dots\}$, while $\mathbb{Z}$ denotes the integers. For $n \in \mathbb{N}$, We use $[n]$ to denote the set $\{1, \dots, n\}$ and $[a, b]$ to denote $\{a, \dots, b\}$. The set of $n \times n$ invertible matrices over $\mathbb{F}$ is denoted by $GL_n(\mathbb{F})$. The set of permutations on $[n]$ is denoted by S_n and the set of cyclic permutations on $[n]$ by C_n. A field is denoted by $\mathbb{F}$, while $\mathbb{F}^\times$ denotes $\mathbb{F}\backslash\{0\}$. We denote the integers mod m by $\mathbb{Z}_m$. The dimension of a vector space V over a field $\mathbb{F}$ is denoted by $\dim_{\mathbb{F}}(V)$. By $\in_r$ we denote uniform random selection from a set, and by "b.b.a to f" we mean black-box access to f.

We denote the set of variables on which HC_n depends as $\mathbf{x} := \{x_{i,j} |\ i, j \in [n], i \neq j\}$, with $|\mathbf{x}| = n^2 - n$ and $X_n = \{x_{i,j} |\ i, j \in [n]\}$ to denote the entire set of n^2 variables on which HC_n is defined. By scaling matrices, we mean diagonal matrices. We treat a scaling matrix $S \in \mathbb{F}^{m \times m}$ as a vector in $\mathbb{F}^m$ by identifying the diagonal as a vector in $\mathbb{F}^m$. Thus, the entries of $S \in \mathbb{F}^{n^2 \times n^2}$, will be referred to as $S_{i,j}$, where $i, j \in [n]$. For a matrix $M \in \mathbb{F}^{m \times m}$ and $S, T \subseteq [m]$, we denote by $M_{\bullet \times T}$ the matrix M restricted to columns in T, by $M_{S \times \bullet}$ the matrix M restricted to rows in S, and by $M_{S \times T}$ the matrix M restricted to the rows and columns in S and T respectively. By $n^{O(1)}$, we denote a polynomially bounded function of n.

2.2 Algebraic and algorithmic preliminaries

▶ **Lemma 14** (Computing Derivatives [33]). *Let $f \in \mathbb{F}[\mathbf{y}]$ be an n-variate degree-d polynomial with bit complexity β. Given black-box access to f, we can obtain in $(nd\beta)^{O(1)}$ time black-box access to a derivative $\frac{\partial f}{\partial y}$ for any $y \in \mathbf{y}$.*

▶ **Definition 15** (Group of Symmetries [18]). *Let $f \in \mathbb{F}[\mathbf{y}]$ be an n-variate polynomial. The group of symmetries of f over $\mathbb{F}$, denoted by $\mathcal{G}_f$, is the set of matrices $\{A \in \mathrm{GL}_n(\mathbb{F}) : f(A\mathbf{y}) = f(\mathbf{y})\}$ which also form a group under matrix multiplication.*

▶ **Definition 16** (Lie Algebra of a polynomial). *Let $f \in \mathbb{F}[\mathbf{y}]$, where $\mathbf{y} := \{y_1, y_2, \ldots, y_n\}$. The Lie Algebra associated with f, denoted by $\mathfrak{g}_f$, is the set $\{A \in \mathbb{F}^{n\times n} \mid \sum_{i,j\in[n]} A_{i,j} y_j \frac{\partial f}{\partial y_i} = 0\}$, which also forms a vector space over $\mathbb{F}$.*

▶ **Lemma 17** (Conjugacy of Lie Algebras [32]). *Let $f \in \mathbb{F}[\mathbf{y}]$ be an n-variate polynomial. If $g(\mathbf{y}) = f(A\mathbf{y})$, where $A \in \mathrm{GL}_n(\mathbb{F})$, then $\mathfrak{g}_g = A^{-1} \cdot \mathfrak{g}_f \cdot A$.*

▶ **Lemma 18** (Computing basis of Lie Algebra [32]). *Given black-box access to an n-variate degree d polynomial $f \in \mathbb{F}[\mathbf{y}]$, a basis of $\mathfrak{g}_f$ can be computed in randomised $(nd\beta)^{O(1)}$ time, where β is the bit complexity of the coefficients.*

For $A \in \mathbb{C}^{n\times n}$, define $e^A := \sum_{i\in\mathbb{N}} \frac{A^i}{i!}$, the sum always converges. Over $\mathbb{C}$, $\mathfrak{g}_f$ is related to $\mathcal{G}_f$ as in Definition 19.

▶ **Definition 19** (Continuous and Discrete Symmetries). *Let $f \in \mathbb{C}[\mathbf{y}]$. If $A \in \mathfrak{g}_f$, then $e^{tA} \in \mathcal{G}_f$ for all $t \in \mathbb{R}$; see [21] for a proof. The continuous symmetries of f are the elements of the set $\{e^{tA} : A \in \mathcal{G}_f \text{ and } t \in \mathbb{R}\}$. All other symmetries in $\mathcal{G}_f$ are the discrete symmetries of f.*

▶ **Definition 20** (Characterisation by circuit identities [15]). *An n-variate polynomial $f(\mathbf{y})$ is said to be characterised by circuit identities if there exist $m = n^{O(1)}$ many polynomials $g_1(\mathbf{z}), \ldots, g_m(\mathbf{z})$, with each g_i computable by a $n^{O(1)}$ size circuit over $\mathbb{Z}$ and $|\mathbf{z}| \le k$, f is the only non-zero polynomial upto scaling that satisfies $g_i(f(\mathbf{y}_1), f(\mathbf{y}_2) \ldots f(\mathbf{y}_k)) = 0$, where $\mathbf{y}_i$ can be computed from $\mathbf{y}$ by a $n^{O(1)}$ size circuit.*

▶ **Definition 21** (Characterisation by symmetries). *A homogeneous degree-d polynomial $f \in \mathbb{F}[\mathbf{y}]$ is characterised by its symmetries if for every homogeneous degree-d polynomial $g \in \mathbb{F}[\mathbf{y}]$, $\mathcal{G}_f \subseteq \mathcal{G}_g$ implies $g = c \cdot f$, for somes $c \in \mathbb{F}^\times$.*

3 Lie Algebra and the Symmetries of HC_n

In Section 3.1, we analyse $\mathfrak{g}_{HC_n}$, the Lie Algebra of HC_n, and show that it comprises diagonal matrices and that it is a $2n-2$ dimensional vector space over any field $\mathbb{F}$, for $n \neq 4$. In Section 3.2, we use $\mathfrak{g}_{HC_n}$ to show that over large enough fields the symmetries of HC_n are generated by permutation and scaling matrices (which proves Theorem 6) and analyse the permutation and scaling symmetries of HC_n. In Section 3.3, we show that HC_n is *not* characterised by its symmetries, proving Theorem 8. In Section 3.4, we show that HC_n is characterised by circuit identities over any field by using the downward self-reducibility of HC_n and prove Theorems 10, 11 and 12. All missing proofs, and the analysis of $\mathfrak{g}_{HC_4}$ and $\mathcal{G}_{HC_4}$ are available in [11].

3.1 Lie Algebra

Proposition 22 shows that $\mathfrak{g}_{HC_n}$ comprises diagonal matrices $A \in \mathbb{F}^{(n^2-n)\times(n^2-n)}$, where for all $\sigma \in C_n$ the sum of the diagonal entries in A corresponding to σ is 0. To prove Proposition 22, we use Observation 23, which follows from Claim 24. Claim 24 states that any two cyclic permutations, when interpreted as Hamiltonian cycles on the complete digraph, must have at least 3 different edges.

▶ **Proposition 22.** *Let $n \ge 3$. Then for $A \in \mathbb{F}^{(n^2-n)\times(n^2-n)}$,*

$$A \in \mathfrak{g}_{HC_n} \iff A \text{ is diagonal and } \sum_{i=1}^{n} A_{(i,\sigma(i)),(i,\sigma(i))} = 0 \text{ for all } \sigma \in C_n.$$

▶ **Observation 23.** *Let $\sigma_1, \sigma_2 \in C_n$ with $\sigma_1 \neq \sigma_2$ and m_σ denote $\prod_{i\in[n]} x_{i,\sigma(i)}$ for $\sigma \in C_n$. Then,*

$$x_{i_1,j_1}\frac{\partial m_{\sigma_1}}{\partial x_{i_2,j_2}} \neq x_{i_3,j_3}\frac{\partial m_{\sigma_2}}{\partial x_{i_4,j_4}},$$

where i_k's, j_k's $\in [n]$, $j_2 = \sigma_1(i_2)$, and $j_4 = \sigma_2(i_4)$ (to ensure non-zero derivatives).

▷ Claim 24. Let $\sigma_1, \sigma_2 \in C_n$ with $\sigma_1 \neq \sigma_2$. There exists $S \subseteq [n]$, such that $|S| = 3$ and $\sigma_1(i) \neq \sigma_2(i)$ for all $i \in S$.

By Proposition 22, any $A \in \mathfrak{g}_{HC_n}$ can be identified with a vector in $\mathbb{F}^{n^2-n}$. Proposition 25 shows $\mathfrak{g}_{HC_n}$ is $2n-2$ dimensional over *all* fields $\mathbb{F}$.

▶ **Proposition 25.** *Let $\mathbb{F}$ be any field and $n = 3$ or $n \geq 5$. Consider $A^{(k)}$, $B^{(\ell)}$ and $C \in \mathbb{F}^{n^2-n}$, where $k \in [2, n], \ell \in [2, n-1]$, defined as:*

$$A^{(k)}_{(i,j)} = \begin{cases} 1 & i = 1, \\ -1 & i = k, \\ 0 & otherwise \end{cases}, \quad B^{(\ell)}_{(i,j)} = \begin{cases} 1 & j = 1, \\ -1 & j = l, \\ 0 & otherwise \end{cases}, \quad C_{(i,j)} = \begin{cases} -1 & i = 2, \\ 1 & j = 2, \\ 0 & otherwise \end{cases}$$

Then, $dim_{\mathbb{F}}(\mathfrak{g}_{HC_n}) = 2n-2$ with $\mathcal{B}_n = \{A^{(2)}, \ldots, A^{(n)}, B^{(2)}, \ldots, B^{(n-1)}, C\}$ as a basis.

To prove Proposition 25, we define a matrix $M^{HC_n} \in \mathbb{F}^{(n-1)!\times(n^2-n)}$ to represent the linear equations given by Proposition 22. Therefore, the nullspace of M^{HC_n} is $\mathfrak{g}_{HC_n}$. The rows of M^{HC_n} are indexed by $\sigma \in C_n$ and columns by variables $x_{i,j}$ in lex order. We then show that the set $\mathcal{B}_n$ in the proposition is linearly independent over *any* $\mathbb{F}$ and that $\mathcal{B}_n \subset \mathfrak{g}_{HC_n}$. Lastly, we construct in $n^{O(1)}$ time a $(n-1)(n-2) \times (n^2-n)$ submatrix $M^{(n)}$ of M^{HC_n} such that $M^{(n)}$ has full row rank over *any* $\mathbb{F}$. Thus, M^{HC_n} has rank at least $(n-1)(n-2)$, which along with the fact that $\mathcal{B}_n \subset \mathfrak{g}_{HC_n}$ proves Proposition 25.

Corollary 26 describes the structure of any $\mathbf{z} \in \mathfrak{g}_{HC_n}$ and follows by considering the matrix formed by the equations in (2), noting that the matrix is full row rank, and that the entries of the elements of $\mathcal{B}_n$ in Proposition 25 satisfy (2). Corollary 27 describes the entries of any continuous symmetry of HC_n, and follows from Corollary 26 and Definition 19. In [11], we further analyse $M^{(n)}$, which proves useful in analysing the scaling symmetries of HC_n over all fields and to prove the correctness of Algorithm 4.

▶ **Corollary 26.** *Let $n = 3$ or $n \geq 5$ and $\mathbf{z} \in \mathbb{F}^{n^2-n}$. Then, $\mathbf{z} \in \mathfrak{g}_{HC_n}$ if and only if the entries $z_{i,j}$ satisfy* (2).

$$\begin{aligned}
z_{i,j} &= z_{i,1} + z_{1,j} + z_{2,3} - z_{1,3} - z_{2,1} \quad i \in [2, n-1],\ j \in [2, n]\ , i \neq j, (i,j) \neq (2,3), \\
z_{n,1} &= (n-2)(z_{1,3} + z_{2,1} - z_{2,3}) - \sum_{\ell=2}^{n} z_{1,\ell} - \sum_{\ell=2}^{n-1} z_{\ell,1}, \\
z_{n,j} &= z_{1,j} + (n-3)(z_{1,3} + z_{2,1} - z_{2,3}) - \sum_{\ell=2}^{n} z_{1,\ell} - \sum_{\ell=2}^{n-1} z_{\ell,1} \\
&= z_{n,1} + z_{1,j} + z_{2,3} - z_{1,3} - z_{2,1}, \quad j \in [2, n-1].
\end{aligned} \tag{2}$$

▶ **Corollary 27.** *Over $\mathbb{C}$, any continuous symmetry $S \in \mathcal{G}_{HC_n}$ is a diagonal matrix, with $S_{i,j}$'s satisfying* (3).

$$S_{i,j} = S_{i,1}S_{1,j}\frac{S_{2,3}}{S_{1,3}S_{2,1}} \quad i \in [2,n-1],\ j \in [2,n]\ , i \neq j, (i,j) \neq (2,3),$$
$$S_{n,1} = \left(\frac{S_{1,3}S_{2,1}}{S_{2,3}}\right)^{n-2}\left(\prod_{\ell=2}^{n} S_{1,\ell}\prod_{\ell=2}^{n-1} S_{\ell,1}\right)^{-1},$$
$$S_{n,j} = S_{n,1}S_{1,j}\frac{S_{2,3}}{S_{1,3}S_{2,1}} \quad j \in [2,n-1]. \tag{3}$$

3.2 The Symmetries of HC_n

Lemma 28 shows that over large enough fields, $\mathfrak{g}_{HC_n}$ contains an element with distinct eigenvalues. Proposition 29 then shows that over such fields, every symmetry of HC_n is generated by permutation and scaling symmetries, proving Theorem 6. The proof of Proposition 29 follows from Lemmas 17 and 28. We then analyse and characterise the permutation and scaling symmetries of HC_n.

▶ **Lemma 28.** *Suppose $|\mathbb{F}| > \binom{n^2-n}{2}$. There exists $A \in \mathfrak{g}_{HC_n}$ such that A has distinct eigenvalues.*

▶ **Proposition 29** (Theorem 6 restated)**.** *Suppose $|\mathbb{F}| > \binom{n^2-n}{2}$. If $A \in \mathcal{G}_{HC_n}$, then $A = PS$, where P is a permutation matrix and S is a scaling matrix.*

Scaling symmetries. Proposition 30 shows that any scaling symmetry S of HC_n satisfies (3) for $n \neq 4$. The proof of Proposition 30 uses the matrix $M^{(n)}$ constructed in the proof of Proposition 22 to show that any scaling symmetry S essentially satisfies a system of equations given by $M^{(n)}$ over $\mathbb{F}^\times$, the solution space of which comprises scaling matrices satisfying (3). In terms of X_n, one can define diagonal matrices L and R from the entries of S, such that $Perm_n(LR) = 1$.

▶ **Proposition 30** (Scaling symmetries are continuous)**.** *Let $\mathbb{F}$ be any field and $n = 3$ or $n \geq 5$. If $S \in \mathcal{G}_{HC_n}$ is a scaling symmetry, then the entries $S_{i,j}$ satisfy* (3).

Permutation Symmetries. Proposition 31 shows that HC_n has non-trivial permutation symmetries. In terms of the matrix X_n, $P^{(\sigma)}$ acts as $X_n \mapsto QX_nQ^T$, where Q is the $n \times n$ matrix corresponding to $\sigma \in S_n$, and $P^{(T)}$ acts as $X_n \mapsto X_n^T$. Proposition 32 shows that the permutation symmetries are generated by those described in Proposition 31. To prove Proposition 32, we use Observation 33, which tells us when the second-order derivatives of HC_n vanish.

▶ **Proposition 31.** *Let $\sigma \in S_n$. The matrices $P^{(\sigma)}$ and $P^{(T)}$, as below, are permutation symmetries of HC_n.*

$$P^{(\sigma)}_{(i,j),(k,\ell)} := \begin{cases} 1 & k = \sigma(i), \ell = \sigma(j), \\ 0 & otherwise \end{cases}, \quad P^{(T)}_{(i,j),(k,\ell)} := \begin{cases} 1 & k = j, \ell = i, \\ 0 & otherwise \end{cases}.$$

Here $i,j,k,\ell \in [n]$, $i \neq j$ and $k \neq \ell$. Also, $P^{(\sigma)}$ and $P^{(T)}$ commute with one another.

▶ **Proposition 32.** *If $P \in \mathcal{G}_{HC_n}$ is a permutation symmetry, then $P = P^{(\sigma)}$ or $P = P^{(\sigma)}P^{(T)}$ for some $\sigma \in S_n$.*

▶ **Observation 33.** *Let $i, j \in [n]$, $i \neq j$ and $R_{i,j}$ be the set of variables $x_{k,\ell}$, other than $x_{i,j}$, such that $\frac{\partial^2 HC_n}{\partial x_{i,j} \partial x_{k,\ell}} = 0$. Then,*

$$x_{k,\ell} \in R_{i,j} \iff i = k, \text{ or } j = \ell, \text{ or } i = \ell \text{ and } j = k.$$

Further, we can partition $R_{i,j}$ as

$$R_{i,j} := Q_{i,j} \sqcup T_{i,j} \sqcup \{x_{j,i}\},\ Q_{i,j} := \{x_{k,j} \mid k \in [n] \backslash \{i, j\}\} \text{ and } T_{i,j} := \{x_{i,k} \mid k \in [n] \backslash \{i, j\}\}.$$

The partitions also satisfy:

1. $\frac{\partial^2 HC_n}{\partial x_{i_1,j_1} \partial x_{i_2,j_2}} \neq 0$ *if* $x_{i_1,j_1} = x_{j,i}$ *and* $x_{i_2,j_2} \in Q_{i,j} \sqcup T_{i,j}$, *or* $x_{i_1,j_1} \in T_{i,j}, x_{i_2,j_2} \in Q_{i,j}$.
2. $\frac{\partial^2 HC_n}{\partial x_{i_1,j_1} \partial x_{i_2,j_2}} = 0$ *if* $x_{i_1,j_1}, x_{i_2,j_2} \in Q_{i,j}$ *or* $x_{i_1,j_1}, x_{i_2,j_2} \in T_{i,j}$.

If $f = HC_n(P\mathbf{x})$, where P is a permutation matrix, then $\frac{\partial^2 f}{\partial x_{i,j} \partial x_{k,\ell}} = 0$ if and only if $P^{-1}(x_{i,j})$ and $P^{-1}(x_{k,\ell})$ are as specified in Observation 33. Thus, P creates a bijection on $R_{i,j}$ which also maps the partitions of $R_{i,j}$ in Observation 33 appropriately. If $P \in \mathcal{G}_{HC_n}$ is a permutation symmetry, then it creates, for each variable $x_{i,j}$, a bijection between the set $R_{i,j}$ and $R_{k,\ell}$, where $x_{k,\ell} = P(x_{i,j})$, such that the bijection also maps the partitions of $R_{i,j}$ as described in Observation 33 accordingly to those of $R_{k,\ell}$. analysing these bijections for variables $x_{1,j}$ and $x_{i,1}$, and noting that the image of $R_{1,j} \cap R_{i,1}$ is a singleton shows how P is generated by the permutation symmetries described in Proposition 31.

3.3 HC_n is not characterised by its symmetries

Proposition 34 shows that for $n \geq 5$, HC_n is *not* characterised by its symmetries, unlike the Permanent polynomial, proving Theorem 8. The proof involves defining a polynomial $g(\mathbf{x})$ as the sum over the image of a monomial, m_σ, under all the permutation symmetries (Proposition 32) of HC_n. Here, $\sigma \in S_n \backslash C_n$ such that $\sigma(i) \neq i$ for all $i \in [n]$. It is easy to see that such a σ exists for $n \geq 5$, for example take $\sigma = (1\ 2)(3\ 4\ \ldots\ n)$. Then, it is not hard to observe that every scaling symmetry of HC_n (Proposition 30) is also a scaling symmetry of g, but $g \neq c \cdot HC_n$ for any $c \in \mathbb{F}^\times$.

▶ **Proposition 34** (Non-characterisation by symmetries). *Let $n \geq 5$ and $\mathbb{F}$ be any field such that $|\mathbb{F}| > \binom{n^2-n}{2}$. There exists a non-zero polynomial f such that $\mathcal{G}_{HC_n} \subseteq \mathcal{G}_f$ but $f \neq c \cdot HC_n$ for any $c \in \mathbb{F}^\times$.*

More generally, in [11], we show how to obtain a scaling symmetry of the Permanent polynomial from that of HC_n. In contrast, we show HC_n, for $n = 3$ and $n = 4$, is characterised by its symmetries over appropriate fields in [11].

3.4 Circuit Identities, Circuit Testing and Flip Theorem

Though HC_n is not characterised by its symmetries, it is characterised by circuit identities (see Definition 20), which follows from HC_n being downward self-reducible, as shown in Lemma 4 of [49] and Lemma 36 here. To prove the circuit identities, we use Lemma 35, an adaptation of Lemma 7.13 in [8], which shows that HC_n can be expressed as HC_m for all $m > n \geq 2$. Lemma 36 then shows that HC_n is downward self-reducible over any field, and can be proved using induction on n. Note that we use the matrix X_n with $x_{i,i}$ replaced by 0's in the statement of Lemmas 35 and 36 for ease of exposition.

▶ **Lemma 35.** *Let $n, m \in \mathbb{N}$ with $m > n \geq 2$. In $O(m^2)$ time, we can construct a $m \times m$ matrix $Y^{(m,n)}$ from X_n such that $HC_m(Y^{(m,n)}) = HC_n(\mathbf{x})$.*

▶ **Lemma 36.** *Let $f \in \mathbb{F}[\mathbf{x}]$. Then, $f = HC_n$ if and only if f satisfies the identities*

$$f(Y^{(n,k)}) = \sum_{i=2}^{k} x_{1,i} f(Y_i^{(n,k-1)}) \quad k \in [3,n], \text{ and } f(Y^{(n,2)}) = x_{1,2}x_{2,1},$$

where $Y^{(n,k)}$ is constructed from X_k as described in the proof of Lemma 35. The matrix $Y_i^{(n,k-1)}$ is obtained from X_k by first swapping row i with row 2 and column i with column 2 in X_k, removing the 1st row and 2nd column to obtain the $(k-1) \times (k-1)$ matrix $X_k^{(i)}$, and constructing $Y^{(n,k-1)}$ using $X_k^{(i)}$ by Lemma 35.

Proof of Theorem 10. Represent the $n-1$ identities in Lemma 36 as polynomials $g_1(\mathbf{x}, \mathbf{z}), \ldots, g_{n-1}(\mathbf{x}, \mathbf{z})$, where $|\mathbf{z}| = n-1$, g_i's are degree 2 polynomials over $\mathbb{Z}$ and have $n^{O(1)}$ size circuit over $\mathbb{Z}$. This establishes circuit identities for HC_n and proves Theorem 10. ◀

Proof of Theorems 11 and 12. Algorithm 1, when given black-box access to a size-s circuit C in $\mathbf{x}$ variables and of degree $d = s^{O(1)}$, tests C on the $n-1$ identities in Lemma 36. The degree of each identity, when we use C in it, is at most d. If C computes HC_n, then Algorithm 1 will always return "Yes". If C does *not* compute HC_n, then some identity fails to hold for C. By the Polynomial Identity Lemma and the fact $|U| > 2d$, the probability that some identity fails to hold is at least $\frac{1}{2}$. By repeating Algorithm 1 $s^{O(1)}$ times, we can boost the success probability to $1 - (\frac{1}{2})^{s^{O(1)}}$. This proves Theorem 11. Theorem 12 follows easily because the random matrices sampled by Algorithm 1 provide the list of counterexamples against polynomial-size circuits, assuming HC_n is not computable by polynomial-size circuits. ◀

Algorithm 1 Circuit Testing for HC_n.

- **Input**: B.b.a to size-s circuit C in $\mathbf{x}$ variables and of degree $s^{O(1)}$.
- **Output**: "Yes" if $\mathrm{C}(\mathbf{x}) = HC_n(\mathbf{x})$, else "No".

1. Let $U \subseteq \mathbb{F}$ with $|U| > 2d$. Choose $n-1$ matrices $M_2, M_3, \ldots, M_n$, where M_i are $i \times i$ matrices with entries chosen uniformly at random from U.
2. For $i \in [2,n]$, check if the i'th identity in Lemma 36 does not hold for C when evaluated at M_i. Return "No" if so.
3. Return "Yes".

4 Equivalence Testing for HC

In this section, we present the ET algorithm for HC_n, and argue its correctness and efficiency by using the structural results proven in Section 3. In Section 4.1, we show the reduction to PS-equivalence; the correctness of the reduction follows from Lemma 28. In Section 4.2, we show how to recover a permutation; the correctness follows from Observation 33. The problem then reduces to checking scaling equivalence with HC_n. In Section 4.3, we show how to recover a scaling matrix by Proposition 30. All missing proofs can be found in [11]. We present the analysis assuming $f = HC_n(A\mathbf{x})$ and that all steps which employ randomisation execute correctly. Every step where randomisation is used has a small probability of failing

due to the assumption on the field size. We run Algorithm 1 at the end to verify the correctness of the recovered transform. If $f \neq HC_n(A\mathbf{x})$, Algorithm 1 will output "No" with high probability. If $f = HC_n(A\mathbf{x})$, Algorithm 1 always accepts.

4.1 Reduction to PS-equivalence

Algorithm 2 shows the reduction to PS-equivalence testing. If $f = HC_n(A\mathbf{x})$, then by Lemma 17, $\dim_{\mathbb{F}}(\mathfrak{g}_f)$ is $2n-2$. We then compute a random element of $\mathfrak{g}_f$ and diagonalise it. The algorithm outputs the diagonalising matrix if it exists. The correctness and complexity analysis of Algorithm 2 is given by Proposition 37.

Algorithm 2 Reduction to PS-equivalence Test.

- **Input**: B.b.a to $f(\mathbf{x})$.
- **Output**: $D \in \mathrm{GL}_{n^2-n}(\mathbb{F})$ s.t. $f(D\mathbf{x}) = HC_n(PS\mathbf{x})$ if $f(\mathbf{x}) = HC_n(A\mathbf{x})$.

1. Compute a basis $\{B_1, B_2, \ldots, B_k\}$ of $\mathfrak{g}_f$ from b.b.a to f. If $k \neq 2n-2$, abort and output "f not equivalent to HC_n" .
2. Let $U \subseteq \mathbb{F}$ be such that $|U| = 3n^5$. Let $a_1, a_2 \ldots a_k \in_r U$ and $C := \sum_{i\in[k]} a_i B_i$. Compute and output a matrix D such that $D^{-1}CD$ is a diagonal matrix. If such a D does not exist, abort output "f not equivalent to HC_n".

▶ **Proposition 37.** *If $f = HC_n(A\mathbf{x})$ for some $A \in GL_{n^2-n}(\mathbb{F})$, with high probability Algorithm 2 computes in randomised $n^{O(1)}$ time a $D \in GL_{n^2-n}(\mathbb{F})$ such that $f(D\mathbf{x}) = HC_n(PS\mathbf{x})$.*

4.2 P-equivalence test

Let $f(\mathbf{x}) = HC_n(P\mathbf{x})$. Using the permutation symmetries and second-order partial derivatives of HC_n, Algorithm 3 recovers a permutation P' such that $f(\mathbf{x}) = HC_n(P'\mathbf{x})$, where P' is P up to a permutation symmetry. Claim 38 and Proposition 39 together argue the correctness by leveraging Observation 33 and the fact that for $f = HC_n(P\mathbf{x})$, where P is a permutation matrix, then $\frac{\partial^2 f}{\partial x_{i,j}\partial x_{k,\ell}} = 0$ if and only if $P^{-1}(x_{i,j})$ and $P^{-1}(x_{k,\ell})$.

▷ Claim 38. Without loss of generality, $P(x_{1,2}) = x_{1,2}$.

▶ **Proposition 39.** *If $f(\mathbf{x}) = HC_n(P\mathbf{x})$, then with high probability Algorithm 3 computes P' in randomised $n^{O(1)}$ time such that $P' = \tilde{P}P$, where $\tilde{P}$ is as described in Proposition 31.*

4.3 S-equivalence test

We assume $n \neq 4$. Suppose $f(\mathbf{x}) = HC_n(S\mathbf{x})$, where $S \in GL_{n^2-n}(\mathbb{F})$ is a scaling matrix. Algorithm 4 gives the scaling equivalence test over $\mathbb{F}_q$. The same algorithm, with some changes, will also work over other fields $\mathbb{F}$, see [11] for details. The correctness of Step 1 follows from Claim 40, while that of the remaining steps follows from Proposition 41. An S-equivalence test for HC_4 over finite fields, $\mathbb{Q}$, and $\mathbb{R}$ is given in [11].

▷ Claim 40. Without loss of generality, $S'_{1,j} = 1$, $S'_{2,3} = 1$ and $S'_{i,1} = 1$, where $i \in [2, n-1], j \in [2, n]$.

▶ **Proposition 41.** *If $f(\mathbf{x}) = HC_n(S\mathbf{x})$, then in deterministic $(n \log q)^{O(1)}$ time Algorithm 4 outputs an S' such that $S'S$ is a scaling symmetry of HC_n.*

Algorithm 3 P-equivalence Test.

- **Input**: B.b.a to $f(\mathbf{x})$.
- **Output**: $P' \in \mathrm{GL}_{n^2-n}(\mathbb{F})$ s.t. $HC_n(P'\mathbf{x}) = f(\mathbf{x})$, if $f(\mathbf{x}) = HC_n(P\mathbf{x})$.

1. Compute b.b.a to $\frac{\partial^2 HC_n}{\partial x_{i,j}\partial x_{k,\ell}}$, where $i \neq j$ and $k \neq \ell$. Check if $\frac{\partial^2 HC_n}{\partial x_{i,j}\partial x_{k,\ell}}$ is identically zero or not, and store this information.
2. Compute the set $R'_{1,2} := \{x_{k,\ell} \mid \frac{\partial^2 HC_n}{\partial x_{1,2}\partial x_{k,\ell}} = 0 \text{ and } (k,\ell) \neq (1,2)\}$. Partition $R'_{1,2}$ as $\{x_{i,j}\} \sqcup Q'_{1,2} \sqcup T'_{1,2}$ such that items 1 and 2 in Observation 33 hold. If no such partition exists, abort and output "f not equivalent to HC_n".
3. Set $P'(x_{1,2}) := x_{1,2}$, $P'(x_{2,1}) := x_{i,j}$ and $P'(x_{1,t}) := x_{i_t,j_t}$, where $x_{i_t,j_t} \in T'_{1,2}$ and $t \in [3,n]$.
4. For all $x_{i_t,j_t} \in T'_{1,2}$, compute R'_{i_t,j_t} as in Step 2 and partition it. Let $\{x_{k_t,\ell_t}\}$ be the singleton set in the partitioning. Set $P'(x_{t,1}) := x_{k_t,\ell_t}$. Abort and output "$f$ not equivalent to HC_n" if the partition does not exist.
5. For $a, b \in [2,n]$, $a \neq b$, compute $R'_{k_a,\ell_a} \cap R'_{i_b,j_b}$, where $P'(x_{a,1}) = x_{k_a,\ell_a}$ and $P'(x_{1,b}) = x_{i_b,j_b}$, as computed in earlier steps. If $R'_{k_a,\ell_a} \cap R'_{i_b,j_b}$ is a singleton $\{x_{c,d}\}$, set $P'(x_{a,b}) := x_{c,d}$, else abort and output "f not equivalent to HC_n".
6. Output P'.

Algorithm 4 S-equivalence test.

- **Input**: B.b.a to $f(\mathbf{x})$.
- **Output**: $S' \in \mathrm{GL}_{n^2-n}(\mathbb{F}_q)$ s.t. $HC_n(S'\mathbf{x}) = f(\mathbf{x})$ and $S' = \tilde{S}S$, where $\tilde{S}$ is a scaling symmetry, if $f(\mathbf{x}) = HC_n(S\mathbf{x})$.

1. Set $S'_{1,j} = 1$, $S'_{2,3} = 1$ and $S'_{i,1} = 1$, where $i \in [2,n-1], j \in [2,n]$.
2. Query f to obtain the coefficients of monomials corresponding to the rows $\sigma_1, \ldots, \sigma_{(n-1)(n-2)}$ of the matrix $M^{(n)}$ constructed in the proof of Proposition 22. Let c_k be the k'th coefficient. If $c_k = 0$ for some k, then abort and ouput "f not equivalent to HC_n".
3. Let $T := \{(i,j) \mid (i,j) \neq (1,k),\ k \in [2,n] \text{ and } (i,j) \neq (\ell,1),\ \ell \in [2,n-1] \text{ and } (i,j) \neq (2,3)\}$. Consider the matrix $M := M^{(n)}_{\bullet \times T}$. Compute $\beta = (det(M))^{-1}$ in $\mathbb{Z}_{q-1}$.
4. For each row σ_k, $k \in [(n-1)(n-2)]$ and $(i,j) \in T$, compute the cofactor of M with respect to σ_k and (i,j), and denote it as $\alpha_{k,(i,j)}$. Set $S'_{i,j} = \prod_{k=1}^{(n-1)(n-2)} c_k^{\beta\alpha_{k,(i,j)} \bmod q-1}$.
5. Output S'.

References

1 Manindra Agrawal and Nitin Saxena. Automorphisms of Finite Rings and Applications to Complexity of Problems. In Volker Diekert and Bruno Durand, editors, *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, volume 3404 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2005. doi:10.1007/978-3-540-31856-9_1.

2 Noga Alon and Ravi B. Boppana. The monotone circuit complexity of boolean functions. *Comb.*, 7(1):1–22, 1987. doi:10.1007/BF02579196.

3 Omkar Baraskar, Agrim Dewan, Chandan Saha, and Pulkit Sinha. NP-Hardness of Testing Equivalence to Sparse Polynomials and to Constant-Support Polynomials. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 16:1–16:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. Full version available at ECCC. doi:10.4230/LIPIcs.ICALP.2024.16.

4 Elwyn R. Berlekamp. Factoring polynomials over large finite fields. In Stanley R. Petrick, Jean E. Sammet, Robert G. Tobey, and Joel Moses, editors, *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation, SYMSAC 1971, Los Angeles, California, USA, March 23-25, 1971*, page 223. ACM, 1971. doi:10.1145/800204.806290.

5 Vishwas Bhargava, Pranjal Dutta, Sumanta Ghosh, and Anamay Tengse. The Complexity of Order-Finding for ROABPs. *CoRR*, abs/2411.18981, 2024. doi:10.48550/arXiv.2411.18981.

6 Lenore Blum, Mike Shub, and Steve Smale. On a Theory of Computation and Complexity over the Real Numbers: NP-completeness, Recursive Functions and Universal Machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.

7 Peter Botta. Linear Transformations that Preserve the Permanent. *Proceedings of the American Mathematical Society*, 18(3):566–569, 1967. URL: http://www.jstor.org/stable/2035500.

8 Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and computation in mathematics*. Springer, 2000.

9 Abhranil Chatterjee, Sumanta Ghosh, Rohit Gurjar, Roshan Raj, and Thomas Thierauf. Bipartite matching is in NC. *Electron. Colloquium Comput. Complex.*, TR26, 2026. URL: https://eccc.weizmann.ac.il/report/2026/100.

10 Richard A. DeMillo and Richard J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Inf. Process. Lett.*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

11 Agrim Dewan. Testing Equivalence to the Hamiltonian Cycle Polynomial, 2026. arXiv:2606.26653.

12 Pranjal Dutta, Mahesh Sreekumar Rajasree, and Santanu Sarkar. Complexity of Monomial Prediction in Cryptography and Machine Learning. In *2024 26th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 118–125, 2024. doi:10.1109/SYNASC65383.2024.00032.

13 Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite Perfect Matching is in Quasi-NC. *SIAM J. Comput.*, 50(3), 2021. Conference version appeared in the proceedings of STOC 2016. doi:10.1137/16M1097870.

14 Ian P. Goulden and David M. Jackson. The Enumeration of Directed Closed Euler Trails and Directed Hamiltonian Circuits by Lagrangian Methods. *Eur. J. Comb.*, 2(2):131–135, 1981. doi:10.1016/S0195-6698(81)80004-2.

15 Joshua A. Grochow. *Symmetry and equivalence relations in classical and geometric complexity theory*. PhD thesis, University of Chicago, Chicago, IL, 2012. URL: https://home.cs.colorado.edu/~jgrochow/grochow-thesis.pdf.

16 Joshua A. Grochow. Monotone Projection Lower Bounds from Extended Formulation Lower Bounds. *Theory Comput.*, 13(1):1–15, 2017. doi:10.4086/TOC.2017.V013A018.

17 Joshua A. Grochow, Ketan D. Mulmuley, and Youming Qiao. Boundaries of VP and VNP. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, Rome, Italy, July 11-15, 2016*, LIPIcs, pages 34:1–34:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.34.

18 Nikhil Gupta and Chandan Saha. On the Symmetries of and Equivalence Test for Design Polynomials. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, Aachen, Germany, August 26-30, 2019*, volume 138 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.53.

19 Nikhil Gupta, Chandan Saha, and Bhargav Thankey. Equivalence Test for Read-Once Arithmetic Formulas. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 4205–4272. SIAM, 2023. doi:10.1137/1.9781611977554.ch162.

20 Leonid Gurvits. Classical complexity and quantum entanglement. *J. Comput. Syst. Sci.*, 69(3):448–484, 2004. doi:10.1016/J.JCSS.2004.06.003.

21 Brian C. Hall. *Lie Groups, Lie Algebras, and Representations*. Springer, 2015.

22 Pavel Hrubes. On Hardness of Multilinearization and VNP-Completeness in Characteristic 2. *ACM Trans. Comput. Theory*, 9(1):1:1–1:14, 2016. doi:10.1145/2940323.

23 Jesko Hüttenhain and Christian Ikenmeyer. Binary determinantal complexity. *Linear Algebra and its Applications*, 504:559–573, 2016. doi:10.1016/j.laa.2016.04.027.

24 Christian Ikenmeyer and Stefan Mengel. On the relative power of reduction notions in arithmetic circuit complexity. *Inf. Process. Lett.*, 130:7–10, 2018. doi:10.1016/J.IPL.2017.09.009.

25 Christian Ikenmeyer and Abhiroop Sanyal. A note on VNP-completeness and border complexity. *Inf. Process. Lett.*, 176:106243, 2022. doi:10.1016/J.IPL.2021.106243.

26 Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004. Conference version appeared in the proceedings of STOC 2001. doi:10.1145/1008731.1008738.

27 Mark Jerrum and Marc Snir. Some Exact Complexity Results for Straight-Line Computations over Semirings. *J. ACM*, 29(3):874–897, 1982. doi:10.1145/322326.322341.

28 Stasys Jukna. Lower Bounds for Tropical Circuits and Dynamic Programs. *Theory Comput. Syst.*, 57(1):160–194, 2015. doi:10.1007/S00224-014-9574-4.

29 Valentine Kabanets and Russell Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Comput. Complex.*, 13(1-2):1–46, 2004. Conference version appeared in the proceedings of STOC 2003. doi:10.1007/S00037-004-0182-6.

30 Erich L. Kaltofen and Barry M. Trager. Computing with Polynomials given by Black Boxes for their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators. *J. Symb. Comput.*, 9(3):301–320, 1990. Conference version appeared in the proceedings of FOCS 1988. doi:10.1016/S0747-7171(08)80015-6.

31 Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1409–1421. SIAM, 2011. doi:10.1137/1.9781611973082.108.

32 Neeraj Kayal. Affine projections of polynomials: extended abstract. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 643–662, 2012. doi:10.1145/2213977.2214036.

33 Neeraj Kayal, Vineet Nair, Chandan Saha, and Sébastien Tavenas. Reconstruction of Full Rank Algebraic Branching Programs. *ACM Trans. Comput. Theory*, 11(1):2:1–2:56, 2019. Conference version appeared in the proceedings of CCC 2017. doi:10.1145/3282427.

34 Pascal Koiran, Natacha Portier, Sébastien Tavenas, and Stéphan Thomassé. A τ -conjecture for Newton Polygons. *Found. Comput. Math.*, 15(1):185–197, 2015. doi:10.1007/S10208-014-9216-X.

35 Joseph M Landsberg. Geometry and complexity theory. *NSF Award Number 1405348. Directorate for Mathematical and Physical Sciences*, 14(1405348):5348, 2015.

36 Arjen K Lenstra, Hendrik W Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. doi:10.1007/BF01457454.

37 Richard J. Lipton. New directions in testing. In Joan Feigenbaum and Michael Merritt, editors, *Distributed Computing And Cryptography, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, October 4-6, 1989*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 191–202. DIMACS/AMS, 1989. doi:10.1090/DIMACS/002/13.

38 Guillaume Malod. *Polynômes et coefficients. (Polynomials and coefficients)*. PhD thesis, Claude Bernard University Lyon 1, France, 2003. URL: https://tel.archives-ouvertes.fr/tel-00087399.

39 Guillaume Malod. The complexity of polynomials and their coefficient functions. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 193–204. IEEE Computer Society, 2007. doi:10.1109/CCC.2007.33.

40 Marvin Marcus and FC May. The Permanent Function. *Canadian Journal of Mathematics*, 14:177–189, 1962. doi:10.4153/CJM-1962-013-4.

41 Ketan Mulmuley. Explicit proofs and the flip. *arXiv preprint arXiv:1009.0246*, 2010. arXiv:1009.0246.

42 Ketan Mulmuley. On P vs. NP and geometric complexity theory: Dedicated to Sri Ramakrishna. *J. ACM*, 58(2):5:1–5:26, 2011. doi:10.1145/1944345.1944346.

43 Ketan Mulmuley and Milind A. Sohoni. Geometric Complexity Theory I: An Approach to the P vs. NP and Related Problems. *SIAM J. Comput.*, 31(2):496–526, 2001. doi:10.1137/S009753970038715X.

44 C. Ramya and Pratik Shastri. On the Hardness of Order Finding and Equivalence Testing for ROABPs. In C. Aiswarya, Ruta Mehta, and Subhajit Roy, editors, *45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2025, BITS Pilani, K K Birla Goa Campus, India, December 17-19, 2025*, LIPIcs, pages 49:1–49:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.FSTTCS.2025.49.

45 Nitin Saxena. *Morphisms of rings and applications to complexity*. PhD thesis, Indian Institute of Technology, Kanpur, 2006. URL: https://www.cse.iitk.ac.in/users/nitin/papers/thesis.pdf.

46 Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

47 Thomas Thierauf. The Isomorphism Problem for Read-Once Branching Programs and Arithmetic Circuits. *Chic. J. Theor. Comput. Sci.*, 1998, 1998. URL: http://cjtcs.cs.uchicago.edu/articles/1998/1/contents.html.

48 Leslie G. Valiant. Completeness classes in algebra. In Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, and Alfred V. Aho, editors, *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261. ACM, 1979. doi:10.1145/800135.804419.

49 Jinshan Zhang and Fengshan Bai. An improved fully polynomial randomized approximation scheme (FPRAS) for counting the number of hamiltonian cycles in dense digraphs. *Theor. Comput. Sci.*, 412(4-5):419–429, 2011. doi:10.1016/J.TCS.2010.10.014.

50 Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. doi:10.1007/3-540-09519-5_73.