# The Complexity of Weighted Boolean #CSP Modulo $k$

**Heng Guo[1], Sangxia Huang[2], Pinyan Lu[3], and Mingji Xia[4]**

1 **University of Wisconsin-Madison**
**Madison, WI 53706, USA**
hguo@cs.wisc.edu
2 **KTH — Royal Institute of Technology**
**Stockholm, Sweden**
sangxia@csc.kth.se
3 **Microsoft Research Asia**
**Beijing, China**
pinyanl@microsoft.com
4 **Institute of Software, Chinese Academy of Sciences**
**Beijing 100190, China**
xmjljx@gmail.com

## Abstract

We prove a complexity dichotomy theorem for counting weighted Boolean CSP modulo $k$ for any positive integer $k > 1$. This generalizes a theorem by Faben for the unweighted setting. In the weighted setting, there are new interesting tractable problems. We first prove a dichotomy theorem for the finite field case where $k$ is a prime. It turns out that the dichotomy theorem for the finite field is very similar to the one for the complex weighted Boolean #CSP, found by [Cai, Lu and Xia, STOC 2009]. Then we further extend the result to an arbitrary integer $k$.

**1998 ACM Subject Classification** F.2 [Theory of Computation] Analysis of Algorithms and Problem Complexity

**Keywords and phrases** #CSP, dichotomy theorem, counting problems, computational complexity

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2011.249

## 1 Introduction

The complexity of counting problems is a fascinating subject. Valiant defined the class #P to capture most of these counting problems [21]. Several other related complexity classes are also well studied. One example is the $\oplus$P class, which consists of language $L$ where there is a polynomial time nondeterministic Turing machine that on input $x \in L$ has an odd number of accepting computations, and on input $x \notin L$ has an even number of accepting computations [20, 18]. This class $\oplus$P can also be formulated as computing the parity of counting problems. In general, for any integer $k$, we may consider the counting problems modulo $k$, and the corresponding complexity class is denoted by $\#_k$P. The class $\oplus$P is in fact $\#_2$P.

Beyond the complexity of individual problems, there has been a great deal of interest in finding complexity dichotomy theorems which state that for a wide class of counting problems, every problem in the class is either computable in polynomial time (tractable) or hard (either NP-hard or #P-hard) [13, 12, 6, 15]. Such dichotomies do not hold without restrictions [17], assuming that the larger complexity class strictly contains P. The restrictions for which

28th Symposium on Theoretical Aspects of Computer Science (STACS'11).
Editors: Thomas Schwentick, Christoph Dürr; pp. 249–260

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


dichotomy theorems are known can be framed in terms of local constraints, most importantly, Constraint Satisfaction Problems (CSP) [19, 10, 3, 4, 5, 11, 9, 14]. In this paper we address weighted #CSP problems, modulo any integer $k$, denoted by $\#_k$CSP.

Here we give a brief description of $\#_k$CSP. Let $\mathscr{F}$ be a set of functions, where each $F \in \mathscr{F}$ is a function mapping Boolean variables to a value. The weighted #CSP problem #CSP($\mathscr{F}$) is defined as follows: The input is a finite set of constraints on Boolean variables $x_1, x_2, \ldots, x_n$ of the form $F(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$, where $F \in \mathscr{F}$. The output is

$$\sum_{x_1,x_2,\ldots,x_n \in \{0,1\}} \prod_{F \in \mathscr{F}} F(x_{i_1}, x_{i_2}, \ldots, x_{i_k}).$$

If each $F$ takes values $0, 1$, then this counts the number of assignments "satisfying" all the Boolean constraints. Generally speaking, functions $F \in \mathscr{F}$ could take arbitrary values. What we consider, $\#_k$CSP, is the case that all $F$ take integer values and the output is computed modulo $k$.

For #CSP, the complexity dichotomy theorem was first obtained for the unweighted case [10], and was later generalized to non-negative values [11]. Cai, Lu and Xia proved a dichotomy theorem for Boolean #CSP, where functions $F \in \mathscr{F}$ take arbitrary complex values [8]. Independently, a dichotomy theorem for real weighted functions was also obtained[2]. In these proofs, there are three extensively used reduction techniques: (1) Gadget construction, (2) polynomial interpolation, and (3) holographic transformation. As pointed out by Valiant [23], for finite fields, holographic transformations and interpolation both appear to offer less flexibility than they do for general counting problems.

There do exist several problems for which counting the number of solutions is #P-complete whereas computing it modulo some integer $k$ is polynomial time computable. One prime example is computing the permanent of a 0/1 matrix, which is #P-complete [20]. The parity version of this problem corresponds to computing the permanent modulo 2, which is the same as the determinant modulo 2, and is therefore computable in polynomial time via linear algebra computations. Some more such tractable parity problems were recently given by Valiant [23]. Furthermore, the characteristic of the finite field may affect the tractability. For example, Valiant showed that $\#_7$Pl-Rtw-Mon-3CNF (counting the number of satisfying assignments of a planar read-twice monotone 3CNF formula, modulo 7) is solvable in P by a holographic algorithm [22], while the parity or general version of the same problem is $\oplus$P-hard or #P-hard, respectively.

These two facts (some useful techniques cannot be adopted in finite fields and there exist some more complicated tractable cases) make it more challenging to obtain a dichotomy for $\#_k$CSP problems. In [14], Faben obtained a dichotomy theorem for unweighted $\#_k$CSP. Essentially, there is no additional tractable case in his dichotomy theorem (except one obvious case). However, when we allow functions to take weights in the ring $\mathbb{Z}_k$, some new non-trivial tractable cases do emerge, which is similar to weighted vs unweighted #CSP without modulus. When moving from unweighted to real or complex weighted cases, the presence of both positive and negative values, and more generally, complex numbers, offers the opportunity of interesting cancelations, which could lead to efficient algorithms. In all such dichotomy theorems, roots of unity plays an essential roles [15, 8, 2, 7]. In finite fields, interesting cancelations do appear and every nonzero element is a root of unity. For general $k$, which may not be a prime, another subtlety is that the computation is performed in a ring $\mathbb{Z}_k$ rather than a field, where some nice property of a field no longer holds.

Our result starts from the finite field case, where the modulus $k$ is an odd prime. In this case, the final result is algebraically the same as the dichotomy for complex weighted #CSP.

The imaginary unit $i = \sqrt{-1}$ plays an important role in the dichotomy for the complex weighted #CSP [8]. Here by "algebraically", we mean that we view $i$ as a fourth primitive root of unit which is also well defined in a finite field (or its extension). Then the dichotomy for $\#_k$CSP is identical to that for complex weighted #CSP. Some of the proof techniques are fairly similar to those in the proof for the complex weighted case [8], while others are completely different. For example, the polynomial interpolation is one of the most important techniques in [8], but it is not available for the finite field.

Hereby we briefly explain why the polynomial interpolation does not work. Consider the simplest case where one would like to realize a unary function $[1, x]$ by polynomial interpolation. The answer to an instance of $\#_p$CSP including $[1, x]$, is a polynomial in the variable $x$. The degree of this polynomial is the number of occurrences of this function $[1, x]$. After replacing all of its occurrences by some realizable unary functions, we can evaluate the polynomial in other points of the variable. Given enough such evaluations, we can get a system of linear equations in the coefficients. The hope is to recover all coefficients by solving this system as long as its not singular. Then we can evaluate the polynomial in the original point $x$. In finite field $\mathbb{Z}_p$, we can reduce the degree of the polynomial to $p - 1$ by Fermat's Little Theorem. So we have $p$ different coefficients to recover. To get a non-singular linear system, we need to evaluate the polynomial on at least $p$ points, which means we need to construct at least $p$ different unary functions. However, in $\mathbb{Z}_p$, there are only $p$ essentially different unary functions of the form $[1, x]$, and thus the interpolation is not even needed if we could construct all of them!

Another difference between the proof here and the one in [8] is that the norm of a complex number is used in [8]. This is an analytical, rather than algebraical, property of complex numbers, and is not available at all in finite fields. Such kinds of similarity and difference between fields with characteristic zero and finite $p$ is one main theme of algebraical geometry [16]. It is interesting to observe similar phenomena in the complexity theory.

For general $k$, let $k = p_1^{r_1} p_2^{r_2} \cdots p_m^{r_m}$, where $p_i$'s are distinct primes, be the prime factorization of $k$. By the Chinese Remainder Theorem, to solve the problem of $\#_k\mathrm{CSP}(\mathscr{F})$ is equivalent to solving all the $\#_{p_i^{r_i}}\mathrm{CSP}(\mathscr{F})$. For $\#_{p^r}$CSP and $p$ being an odd prime, we prove a surprising result that $\#_{p^r}\mathrm{CSP}(\mathscr{F})$ is tractable iff $\#_p\mathrm{CSP}(\mathscr{F})$ is, assuming #P is not equal to P. One direction is trivial, namely if $\#_{p^r}\mathrm{CSP}(\mathscr{F})$ can be solved in polynomial time, so can $\#_p\mathrm{CSP}(\mathscr{F})$. The reduction in the other direction is not of the black box style. We need the dichotomy for $\#_p\mathrm{CSP}(\mathscr{F})$ to state all the tractable cases, assuming #P is not equal to P, and we also need to explicitly use algorithms to solve such tractable cases. The algorithm for $\#_{p^r}\mathrm{CSP}(\mathscr{F})$ has a time complexity which is $n^r$ times larger than that of the algorithm for $\#_p\mathrm{CSP}(\mathscr{F})$. We use a different treatment to solve the case that $p = 2$.

## 2 Preliminaries

Let $k$ be a given constant integer. In this paper we address the following type of counting problems, called weighted Boolean $\#_k$CSP. Let $\mathscr{F}$ be a set of functions, where each $f \in \mathscr{F}$ is a function $f : \{0, 1\}^r \to \mathbb{Z}$, mapping Boolean variables to integers. We call $r$ the *arity* of $f$. The problem $\#_k\mathrm{CSP}(\mathscr{F})$ is defined as follows: The input is a finite set of constraints on Boolean variables $x_1, x_2, \ldots, x_n$ of the form $f_j(x_{i_{j,1}}, x_{i_{j,2}}, \ldots, x_{i_{j,r_j}})$, where $f_j \in \mathscr{F}$. The output is

$$\left( \sum_{x_1, x_2, \ldots, x_n \in \{0,1\}} \prod_j f_j(x_{i_{j,1}}, x_{i_{j,2}}, \ldots, x_{i_{j,r_j}}) \right) \bmod k. \tag{1}$$

Since we are only interested in the final value modulus $k$, it is equivalent to view that all the functions take values in the ring $\mathbb{Z}_k$.

A *symmetric* function $f$ of arity $r$ on Boolean variables can be expressed by $[f_0, f_1, \ldots, f_r]$, where $f_j$ is the value of $f$ on inputs of weight $j$. We also use $\Delta_0, \Delta_1$ to denote $[1, 0]$ and $[0, 1]$ respectively. A binary function $f$ is also expressed by the matrix $\begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix}$.

Suppose $f$ is a function on input variables $x_1, x_2, \ldots, x_r$. $f^{x_s=c}$ denotes the function $f^{x_s=c}(x_1, \ldots, x_{s-1}, x_{s+1}, \ldots, x_r) = f(x_1, \ldots, x_{s-1}, c, x_{s+1}, \ldots, x_r)$, and $f^{x_s=*}$ denotes the function $f^{x_s=*} = f^{x_s=0} + f^{x_s=1}$.

The underlying relation of $f$ is given by $R_f = \{X \in \{0,1\}^r | f(X) \neq 0\}$. We also view relations as functions from $\{0,1\}^r$ to $\{0,1\}$. In this way, $R_f$ could be viewed as *the unweighted version* of $f$. If the modulus $k$ is a prime, we could copy $f$ $k-1$ times to get $f^{k-1}$ which, by Fermat's Little Theorem, is the unweighted version of it. In this way, we would be able to use some existing results for unweighted $\#_{\mathrm{k}}$CSP problems.

A relation $R \subseteq \{0,1\}^r$ being *affine* means it is the affine linear subspace composed of solutions of a system of affine linear equations, equivalently, if $a, b, c \in R$, then $a \oplus b \oplus c \in R$. If $R_f$ is affine, we say $f$ has affine support.

One important starting point of our work are the hardness results for unweighted $\#_{\mathrm{k}}$CSP [14]. For the unweighted case, every function $f$ takes 1 if the input $X \in R_f$, and takes 0 otherwise.

▶ **Theorem 1.** [14] *Given an unweighted function set $\mathscr{F}$, and an integer $k$, $\#_{\mathrm{k}}\mathrm{CSP}(\mathscr{F})$ is computable in polynomial time if all the relations in $\mathscr{F}$ are affine, or if $k = 2$ and all functions in $\mathscr{F}$ are closed under complement. Otherwise it is $\#_k$P-hard* [1].

As a corollary, we have the following hardness result.

▶ **Corollary 2.** *$\#_{\mathrm{k}}\mathrm{CSP}(\{[0,1,1]\})$ and $\#_{\mathrm{k}}\mathrm{CSP}(\{[1,1,0]\})$ are $\#_k$P-hard for all $k$.*

We also need the following Pinning Lemma for $\#_{\mathrm{k}}$CSP.

▶ **Lemma 3.** *For every $\mathscr{F}$ and odd prime $k$, $\#_{\mathrm{k}}\mathrm{CSP}(\mathscr{F} \cup \{[1,0],[0,1]\}) \leq_T \#_{\mathrm{k}}\mathrm{CSP}(\mathscr{F})$.*

The proof is similar to that in [2].

We regard a function $f$ and $c \cdot f$ as the same function, where $c$ is a constant relatively prime to the modulus $k$. Our study on $\#_{\mathrm{k}}\mathrm{CSP}(\mathscr{F})$ starts with prime modulus. Doing computation modulo a prime is similar to computing with complex numbers in many aspects. For a given $k$, we define $i_k$ as an element that satisfies $i_k^2 \equiv -1 \pmod{k}$. In some circumstances, $i_k$ is an element of $\mathbb{Z}_k$, while in other situations, we need to extend the field and consider $\mathbb{Z}_k[x]/(x^2+1)$. There are essentially two elements satisfying this property, but it doesn't matter which one we pick as $i_k$.

We further define two classes of functions, for which the $\#_{\mathrm{k}}$CSP problems are tractable. Let $X$ be an $r+1$ dimensional column vector $(x_1, x_2, \ldots, x_r, 1)$ over Boolean field $\mathbb{F}_2$. Suppose $A$ is a Boolean matrix. $\chi_{AX}$ denotes the affine relation on inputs $x_1, x_2, \ldots, x_r$, whose value is 1 if $AX$ is the zero vector, 0 if $AX$ is not the zero vector.

$\mathscr{A}_k$ denotes all functions which have the form $\chi_{AX} i_k^{L_1(X)+L_2(X)+\cdots+L_n(X)}$ in modulo $k$, where $L_j$ is a 0-1 indicator function $\chi_{\langle \alpha_j, X \rangle}$, $\alpha_j$ is a $r+1$ dimensional vector, and the inner

[1] We keep the statement as in Faben's paper. Technically, in the case that $k = 2k'$ where $k' > 1$ is odd, and $\mathscr{F}$ are closed under complement and not all affine, we believe that we can only claim the problem is $\#_{k'}$P-hard.

product $\langle \cdot, \cdot \rangle$ is over $\mathbb{Z}_2$. The additions among $L_j(X)$ are just the usual addition in $\mathbb{Z}$. Since it is the power of $i$, it can be computed modulo 4, but not modulo 2. (Since we ignore the global constant, all functions that are constant multiples of these functions are also in this class.)

$\mathscr{P}_k$ denotes the class of functions which, in modulo $k$, can be expressed as a product of unary functions, binary equality function ($[1, 0, 1]$), and binary disequality function ($[0, 1, 0]$).

It is often useful to view a $\#_k$CSP instance as a bipartite graph $G = (U, V, E)$ where $U$ corresponds to the set of constraints and $V$ corresponds to the set of variables. Edge $(u, v) \in E$ iff variable $v$ appears in constraint $u$. A subgraph of $G$ is simply a certain combination of constraints in terms of CSP, and is sometimes called gadget. It is easy to see that if there are several connected components in $G$, then the result of the whole instance is exactly the product of that in the connected components. Therefore, it is sufficient to consider connected $\#_k$CSP instances.

We also need some knowledge from number theory to deal with prime powers. Given $k$, $a$ is a quadratic residue modulo $k$ if there exists $y$ such that $y^2 \equiv a \pmod{k}$. Thus, $i_k$ exists in $\mathbb{Z}_k$ if and only if $-1$ is a quadratic residue modulo $k$.

▶ **Lemma 4.** *Let $p$ be an odd prime and $k = p^r$. $-1$ is a quadratic residue modulo $p$ if and only if it is a quadratic residue modulo $k$.*

**Proof.** The "if" direction is obvious. If there exist some $i_p \in [p]$ that $i_p^2 \equiv -1 \pmod{p}$, we consider the number $j_t = i_p + tp$, where $t$ is an integer ranging from 0 to $p^{r-1} - 1$. If there exist $t$ and $t'$ that $j_t^2 \equiv j_{t'}^2 \pmod{p^r}$, it is easy to compute that $p^{r-1} | (p(t+t') + 2i_p)(t-t')$, because $|t - t'| < p^{r-1}$, $p | p(t + t') + 2i_p$. We get $p | 2i_p$, which is impossible. Thus the $p^{r-1}$ values $\{j_t^2 \pmod{p^r}\}$ are distinct and there must exist some $t$ such that $j_t^2 \equiv -1 \pmod{p^r}$. ◀

## 3 Complexity in the finite field $\mathbb{Z}_p$

In this section, we deal with the complexity of counting CSP problems in the finite field $\mathbb{Z}_p$. The parity case that $p = 2$, which is in fact the same as the unweighted case, has been solved in [14] (Theorem 1). In the following we always assume that $p$ is an odd prime. All computations are done in the finite field $\mathbb{Z}_p$. For convenience, we often use the usual notation $=$ instead of $\equiv \pmod{p}$.

The counting CSP problem for $\mathscr{P}_p$ or $\mathscr{A}_p$ is tractable. The algorithm for $\mathscr{P}_p$ is based on decomposing functions into separated components that is easy to solve. The algorithm for $\mathscr{A}_p$ is similar to that for the complex weighted #CSP problems. We need the following two lemmas. The proof for the modulo case here is similar to the complex weighted case in [8].

▶ **Lemma 5.** *Let $F(x_1, x_2, \ldots, x_k) = \chi_{AX} i^{L_1(X)+L_2(X)+\cdots+L_n(X)} \in \mathscr{A}$. If $AX = 0$ is infeasible over $\mathbb{Z}_2$, then $\sum_{x_1,x_2,\ldots,x_k} F = 0$. If $AX = 0$ is feasible, then in polynomial time, we can construct another function $H(y_1, y_2, \ldots, y_s) = i^{L'_1(Y)+L'_2(Y)+\cdots+L'_n(Y)} \in \mathscr{A}$, such that $0 \le s \le k$, and $\sum_{x_1,x_2,\ldots,x_k} F = \sum_{y_1,y_2,\ldots,y_s} H$.*

▶ **Lemma 6.** *Let $F(x_1, x_2, \ldots, x_k) = i^{L_1(X)+L_2(X)+\cdots+L_n(X)}$. Exactly one of the following two statements hold:*

1. (Congruity) *There exists a constant $c \in \{1, -1, i, -i\}$ such that for all $x_2, x_3, \ldots, x_k \in \{0, 1\}$ we have $F^{x_1=1}/F^{x_1=0}(x_2, x_3, \ldots, x_k) = c$;*
2. (Semi-congruity) *There exists a constant $c \in \{1, i\}$ and an affine subspace $S$ of dimension $k - 2$ on $T = \{(x_2, x_3, \ldots, x_k) \mid x_i \in \mathbb{Z}_2\}$, such that $F^{x_1=1}/F^{x_1=0}(x_2, x_3, \ldots, x_k) = c$ on $S$, and $F^{x_1=1}/F^{x_1=0}(x_2, x_3, \ldots, x_k) = -c$ on $T - S$.*

We first apply Lemma 5 to get rid of the $\chi_{AX}$ factor, and then use the congruity or semi-congruity property of constraint functions proved in Lemma 6 to eliminate variables one by one. Details can be found in [8].

We will then show that for any other functions the problem is hard. Before the hardness proof we will mention two constructions, which will be used throughout our proofs.

Given a function $f$ and any positive integer $k$, we can simulate a function $g$ such that any entry of $g$ is the corresponding entry of $f$ to the $k$th power. This is done by connecting corresponding edges of $k$ copies of $f$ with an equality of arity $k+1$. Figure 1 is a simple example when $k=2$ and the arity of $f$ is 3. The other construction is for binary functions. Given two binary functions $f$ and $g$, whose matrices are $F$ and $G$ respectively, we connect them directly via a binary equality as shown in Figure 2. It is easy to check that the matrix of the resulting function is $FG$.

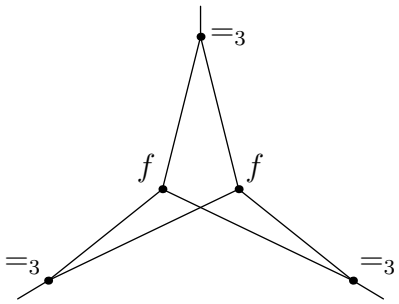


**Figure 1** Duplicate two copies of $f$

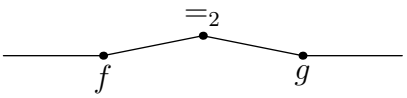


**Figure 2** Directly connect two binary functions

The starting point of our hardness proof is the following lemma. In the rest of this section we may omit the subscripts of $\mathscr{A}_p$ and $\mathscr{P}_p$ when it is clear from context.

▶ **Lemma 7.** *If* $[a,b,c] \notin \mathscr{A} \cup \mathscr{P}$, $\#_{\mathrm{p}}\mathrm{CSP}(\{[a,b,c]\})$ *is* $\#_p$*P-hard. To be explicit, all tractable functions* $[a,b,c]$ *from* $\mathscr{A} \cup \mathscr{P}$ *have one of the following forms:* $[x,0,y]$, $[0,x,0]$, $[x^2,xy,y^2]$, $x[1,\pm i,1]$ *or* $x[1,\pm 1,-1]$.

This lemma says, if restricted to one single symmetric binary function, a dichotomy theorem holds. The same lemma also served as the hardness starting point for the complex weighted dichotomy [8]. However, the proof techniques are completely different. The main proof tool for the complex weighted dichotomy [8] is polynomial interpolation, which is not available here as was explained in Section 1. Before proving this lemma, we state several useful facts.

▶ **Lemma 8.** *For any symmetric binary function* $[0,b,c]$ *and a prime number* $p$*, where* $bc \not\equiv 0 \pmod p$, $\#_{\mathrm{p}}\mathrm{CSP}(\{[0,b,c]\})$ *is* $\#_p$*P-hard.*

**Proof.** Via the construction mentioned above, we can realize $[0,b^k,c^k]$. Taking $k=p-1$, by Fermat's Little Theorem, it becomes $[0,1,1]$. By Corollary 2, the $\#_p\mathrm{CSP}$ problem is $\#_p$P-hard. ◀

We also need the following lemma to realize new binary functions. The new binary functions are not by an explicit construction but an existent argument, which crucially uses the finiteness of the field.

▶ **Lemma 9.** *For any non-degenerate $2 \times 2$ matrix $A$ in $\mathbb{Z}_p$, there exists $k$ such that $A^k \equiv I \pmod p$ where $I$ is the identity matrix.*

**Proof.** Since $\mathbb{Z}_p$ is finite, there are finitely many non-degenerate $2 \times 2$ matrices. Thus, by Pigeonhole Principle, there exists $p$ and $q$ such that $p < q$ and $A^p \equiv A^q \pmod p$. Taking the smallest such pair and letting $k = q - p$, it is easy to see that $A^k \equiv I \pmod p$. ◀

▶ **Corollary 10.** *For any non-degenerate $2 \times 2$ matrix $A$ in $\mathbb{Z}_p$, there exists a positive integer $k$ such that $A^k \equiv A^{-1} \pmod p$.*

▶ **Lemma 11.** *Let $F$ be a function of matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, where $p \nmid abcd$ and $a^2d^2 \not\equiv b^2c^2 \pmod p$. $\#_{\mathrm{p}}\mathrm{CSP}(\{F\})$ is $\#_p$P-hard.*

**Proof.** We can realize $\begin{bmatrix} a^2 & c^2 \\ b^2 & d^2 \end{bmatrix}$ by two copies of the function. Since $a^2d^2 \neq b^2c^2$, this matrix is non-degenerate. Thus by Corollary 10, we can realize $(a^2d^2-b^2c^2)^{-1}\begin{bmatrix} d^2 & -c^2 \\ -b^2 & a^2 \end{bmatrix}$. As we consider the problem in the field $\mathbb{Z}_p$, $(a^2d^2 - b^2c^2)^{-1}$ is just a constant factor and we may ignore it. By the pinning Lemma 3, we can realize $[d^2, -c^2]$, and hence $[d^2, 0, -c^2]$, by connecting it to a $=_3$. Then the following function

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} d^2 & 0 \\ 0 & -c^2 \end{bmatrix}\begin{bmatrix} a & c \\ b & d \end{bmatrix} = \begin{bmatrix} a^2d^2 - b^2c^2 & acd^2 - bc^2d \\ acd^2 - bc^2d & 0 \end{bmatrix},$$

or $[a^2d^2 - b^2c^2, cd(ad - bc), 0]$ is realizable. Since $p \nmid abcd$ and $a^2d^2 \neq b^2c^2$, we have $a^2d^2 - b^2c^2 \neq 0$ and $cd(ad - bc) \neq 0$. By Lemma 8, $\#_{\mathrm{p}}\mathrm{CSP}(\{F\})$ is $\#_p$P-hard. ◀

Now we can prove Lemma 7.

**Proof.** (Lemma 7) If $a = 0$, we know $bc \neq 0$, otherwise it is in one of the five exceptional cases. So by Lemma 8, $\#_{\mathrm{p}}\mathrm{CSP}(\{[a, b, c]\})$ is $\#_p$P-hard . The case $c = 0$ is symmetric. Since $[a, b, c] \notin \mathscr{A} \cup \mathscr{P}$, we know $b \neq 0$. Therefore we will assume in the following that $abc \neq 0$.

By Lemma 11, $\#_{\mathrm{p}}\mathrm{CSP}(\{[a, b, c]\})$ is $\#_p$P-hard if $b^4 \neq a^2c^2$. Moreover, if $b^2 = ac$, then $[a, b, c] \in \mathscr{P}$. Therefore in the following, we assume that $b^2 = -ac$. Since $[a, b, c] \notin \mathscr{A} \cup \mathscr{P}$, we must have that $a \neq \pm c$.

Next we connect two copies of $[a, b, c]$ to realize $[a^2 + b^2, ab + bc, b^2 + c^2]$. Since $b^2 = -ac$, we actually have $[a(a - c), b(a + c), c(c - a)] \triangleq [a', b', c']$. It is easy to verify that $b'^4 \neq a'^2c'^2$, and thus $\#_{\mathrm{p}}\mathrm{CSP}(\{[a, b, c]\}) \geq_T \#_{\mathrm{p}}\mathrm{CSP}(\{[a', b', c']\})$, which is $\#_p$P-hard . ◀

▶ **Lemma 12.** *If $R_F$ is not affine, then $\#_{\mathrm{p}}\mathrm{CSP}(\{F\})$ is $\#_p$P-hard.*

**Proof.** We can easily reduce the unweighted case to the weighted one, the hardness follows. ◀

Now we come to the two key lemmas for the hardness proof. Both proofs inductively reduce the arity of a function. Suppose $\mathscr{F} \not\subseteq \mathscr{A}$ and $\mathscr{F} \not\subseteq \mathscr{P}$. Thus there exists $F \notin \mathscr{A}$ and $G \notin \mathscr{P}$, where $F, G \in \mathscr{F}$. (It is possible that $G = F$). From $F$ and $G$, we recursively simulate functions with smaller arity, keeping the property of being not in $\mathscr{A}$ and not in $\mathscr{P}$ respectively. The proofs of these two lemmas are very similar to those in [8]. Due to space limitation, we give a sketch of proof for Lemma 14 in the Appendix and omit the proof for Lemma 13.

▶ **Lemma 13.** *If $F \notin \mathscr{A}$, then either $\#_{\mathrm{p}}\mathrm{CSP}(\{F\})$ is $\#_p$P-hard, or we can simulate a unary function $H \notin \mathscr{A}$, that is, there is a reduction from $\#_{\mathrm{p}}\mathrm{CSP}(\{F, H\})$ to $\#_{\mathrm{p}}\mathrm{CSP}(\{F\})$.*

▶ **Lemma 14.** *For any function $F \notin \mathscr{P}$, either $\#_{\mathrm{p}}\mathrm{CSP}(\{F\})$ is $\#_p$P-hard , or we can simulate, using $F$, a function $[a, 0, 1, 0]$ (or $[0, 1, 0, a]$), where $a \neq 0$, or a binary function $H \notin \mathscr{P}$ having no zero values.*

Now we are ready to prove the main lemma.

▶ **Lemma 15.** *Let $p$ be an odd prime, and $\mathscr{F}$ a class of functions mapping Boolean inputs to $[p]$. If $\mathscr{F} \subseteq \mathscr{A}$ or $\mathscr{F} \subseteq \mathscr{P}$, $\#_{\mathrm{p}}\mathrm{CSP}(\mathscr{F})$ is computable in polynomial time. Otherwise, $\#_{\mathrm{p}}\mathrm{CSP}(\mathscr{F})$ is $\#_p$P-hard .*

**Proof.** For $\mathscr{A}$ and $\mathscr{P}$, their polynomial time algorithms are given above.

If $\mathscr{F} \not\subseteq \mathscr{A}$ and $\mathscr{F} \not\subseteq \mathscr{P}$, by Lemma 13, either $\#_{\mathrm{p}}\mathrm{CSP}(\mathscr{F})$ is $\#_p$P-hard , or we can simulate a function $F = [1, \lambda] \notin \mathscr{A}$. In particular $\lambda \notin \{0, \pm 1, \pm i\}$. By Lemma 14, either $\#_{\mathrm{p}}\mathrm{CSP}(\mathscr{F})$ is $\#_p$P-hard , or we can simulate a function $P = [a, 0, 1, 0]$, or $P' = [0, 1, 0, a]$, where $a \neq 0$, or a binary function $H \notin \mathscr{P}$ having no zero values.

Firstly, we prove $\#_{\mathrm{p}}\mathrm{CSP}(\{F, P\})$ is $\#_p$P-hard . Clearly $P^{x_1=*} = [a, 1, 1]$. If $a \notin \{1, -1\}$, it is $\#_p$P-hard by Lemma 7. If $a \in \{1, -1\}$, we can construct $Q(x_1, x_2) = \sum_{x_3} P(x_1, x_2, x_3)F(x_3) = [a, \lambda, 1]$, which is $[\pm 1, \lambda, 1]$. Both of them are $\#_p$P-hard by Lemma 7. The proof for $\#_{\mathrm{p}}\mathrm{CSP}(\{F, P'\})$ is the same.

Secondly, we prove $\#_{\mathrm{p}}\mathrm{CSP}(\{F, H\})$ is $\#_p$P-hard . After normalizing, we may suppose $H = \begin{bmatrix} 1 & x \\ y & z \end{bmatrix}$, where $xyz \neq 0$, and $z \neq xy$. There are two cases, depending on whether $z = -xy$.

For the case $z \neq -xy$, we conclude that it is hard by applying Lemma 11 on $H$.

For the case $z = -xy$, we construct some binary functions with an integer parameter $s$ as follows:

$$\begin{aligned} \sum_{x_3} H(x_1, x_3)H(x_2, x_3)(F(x_3))^s &= [1 + \lambda^s x^2, (y + \lambda^s xz), (y^2 + \lambda^s z^2)] \\ &= [1 + \lambda^s x^2, y(1 - \lambda^s x^2), y^2(1 + \lambda^s x^2)]. \end{aligned}$$

As $\lambda$ is not a power of $i$, at most one of the two values $x^2$ and $\lambda x^2$ can be a power of $i$. Now we choose $s = 0$ or $s = 1$ above so that $\lambda^s x^2 \notin \{\pm 1, \pm i\}$.

After normalizing, we may write the function $[1 + \lambda^s x^2, y(1 - \lambda^s x^2), y^2(1 + \lambda^s x^2)]$ as $[1, y(1 - \lambda^s x^2)(1 + \lambda^s x^2)^{-1}, y^2]$, noticing that $1 + \lambda^s x^2 \neq 0$. We claim that this function is not one of the five tractable cases from Lemma 7. Since there are no zero entries, clearly it is not the first two cases. It has rank 2, therefore it is not the third case. If it were the fourth tractable case $[1, \pm i, 1]$, then $y = \pm 1$, and $(1 - \lambda^s x^2)(1 + \lambda^s x^2)^{-1} = \pm i$. This implies that $\lambda^s x^2 = \pm i$, which is impossible. If $[1, y(1 - \lambda^s x^2)(1 + \lambda^s x^2)^{-1}, y^2] = [1, \pm 1, -1]$, the fifth tractable case, then $y = \pm i$, and again $(1 - \lambda^s x^2)(1 + \lambda^s x^2)^{-1} = \pm i$, also impossible. ◀

## 4 Dichotomy for a general integer $k$

We first deal with the case when $k$ is a power of an odd prime. Then we use Chinese Remainder Theorem to prove hardness for other composite numbers. Beigel and Gill have shown that the class $\#_{p^r}P$ is the same as $\#_pP$ and for a composite $k$ having two or more prime factors, the modulo counting class is a union of counting classes modulo each of its prime factors [1]. Therefore, we only talk about $\#_p$P-hard for prime $p$.

▶ **Lemma 16.** *Suppose $p$ is an odd prime, $k = p^r$ for some integer $r \geq 1$. $\mathscr{F}$ is a finite set of constraint functions. If, after taking modulus $p$, $\mathscr{F} \subseteq \mathscr{A}_p$ or $\mathscr{F} \subseteq \mathscr{P}_p$, $\#_{\mathrm{k}}\mathrm{CSP}(\mathscr{F})$ has polynomial time algorithms. Otherwise, $\#_{\mathrm{k}}\mathrm{CSP}(\mathscr{F})$ is $\#_p$P-hard .*

**Proof.** If $\#_{\mathrm{p}}\mathrm{CSP}(\mathscr{F})$ is $\#_p$P-hard, then $\#_{\mathrm{k}}\mathrm{CSP}(\mathscr{F})$ must be $\#_p$P-hard. Therefore we only need to show the tractable part. We decompose every function $f \in \mathscr{F}$ into the sum of two functions $g_f$ and $h_f$, such that all values of $h_f$ are multiples of $p$, and $g_f \in \mathscr{P}_k$ or $g_f \in \mathscr{A}_k$, depending on whether $f \in \mathscr{P}_p$ or $f \in \mathscr{A}_p$. Such a decomposition is always possible. If $f \in \mathscr{P}_p$, assuming that $f = \prod f_i$, where $f_i$ is either unary, or binary equality or disequality, the we can simply take $g_f = \prod f_i$ in modulo $k$. Since $g_f \equiv f \pmod{p}$, we can see that values of $h_f = f - g_f$ are multiples of $p$. On the other hand, if $f \in \mathscr{A}_p$, then $f$ could be expressed as $f = \chi_{AX} i_p^{\sum L_i(X)}$. Let $g_f = \chi_{AX} i_k^{\sum L_i(X)}$. It is easy to see that $g_f \equiv f \pmod{p}$, and $h_f$ satisfies our condition.

Given the decomposition, we can express the final summation in the following way

$$\begin{aligned}
&\sum_{x_1,x_2,\ldots,x_n\in\{0,1\}} \prod f(x_{i_1}, x_{i_2}, \ldots, x_{i_r}) \bmod k \\
= &\sum_{x_1,x_2,\ldots,x_n\in\{0,1\}} \prod (g_f(x_{i_1}, x_{i_2}, \ldots, x_{i_r}) + h_f(x_{i_1}, x_{i_2}, \ldots, x_{i_r})) \bmod k \\
= &\sum_{f'_1\in\{g_{f_1},h_{f_1}\}} \cdots \sum_{f'_n\in\{g_{f_n},h_{f_n}\}} \left( \sum_{x_1,x_2,\ldots,x_n\in\{0,1\}} \prod f'_i(x_{i_1}, x_{i_2}, \ldots, x_{i_r}) \right) \bmod k
\end{aligned}$$

We only need to consider the assignments of functions such that the summation in the parenthesis is nonzero. To ensure that this is nonzero, no more than $r$ of the functions $f'_i$ can be assigned $h_{f_i}$. The total number of such combinations is of order $O(n^{r+1})$.

For every such combination, assume that $f'_1, \ldots, f'_r$ are assigned. Since $\mathscr{F}$ is finite, the degree of the functions is bounded. Therefore, constantly many variables are involved in $f'_1, \ldots, f'_r$. We can list all assignments to these variables in constant time. We can express each assignment with the help of Lemma 3, and obtain a new instance in $\#_{\mathrm{k}}\mathrm{CSP}(\mathscr{F}')$ such that $\mathscr{F}' \subseteq \mathscr{A}$ or $\mathscr{F}' \subseteq \mathscr{P}$, depending on the case of $\mathscr{F}$. Therefore, we can compute the value of these instances in polynomial time, and thus we can compute the value of the whole instance efficiently. ◀

Now we deal with $k = 2^r$. We need the following claim to establish the connection between the weighted and unweighted case.

▶ **Lemma 17.** *For any positive integer $r$ and $t$, $(1 + 2t)^{2^r} \equiv 1 \pmod{2^r}$.*

Therefore, we only need to duplicate the weighted functions $k = 2^r$ times to obtain an unweighted function. Note this process actually converts all odd values to 1, and all even values to 0. Then we have the following result for $k = 2^r$.

▶ **Lemma 18.** *If $k = 2^r$ and $r > 1$, then $\#_{\mathrm{k}}\mathrm{CSP}(\mathscr{F})$ is $\#_2$P-hard, unless all functions in $\mathscr{F}$ are affine modulo 2, for which we have a polynomial time algorithm.*

**Proof.** Hardness can be proved by considering the unweighted version of $\mathscr{F}$ and applying Theorem 1. Algorithm for an affine $\mathscr{F}$ is similar to that in Lemma 16, except that for a given combination and assignment, we calculate the value of the gadget directly instead of applying the Pinning Lemma. This can be done efficiently according to [8]. ◀

Based on Lemma 15, Lemma 16, Lemma 18 and the Chinese Remainder Theorem, we conclude with our main result:

▶ **Theorem 19.** *Let $k = 2^{r_0} p_1^{r_1} p_2^{r_2} \cdots p_m^{r_m}$, where $p_i$'s are distinct odd primes, $r_0 \geq 0$, and $r_i \geq 1$ for $i = 1, 2, ..., m$. Let $\mathscr{F}$ be a set of functions. $\#_k CSP(\mathscr{F})$ is in P if one of the following three conditions is satisfied.*

1. *$r_0 = 0$. $\mathscr{F} \subseteq \mathscr{A}_{p_i}$ or $\mathscr{F} \subseteq \mathscr{P}_{p_i}$ for all $i \in [m]$.*
2. *$r_0 = 1$. $\mathscr{F} \subseteq \mathscr{A}_2$ or every function in $\mathscr{F}$ are closed under complement after mod 2. $\mathscr{F} \subseteq \mathscr{A}_{p_i}$ or $\mathscr{F} \subseteq \mathscr{P}_{p_i}$ for all $i \in [m]$.*
3. *$r_0 \geq 2$. $\mathscr{F} \subseteq \mathscr{A}_2$. $\mathscr{F} \subseteq \mathscr{A}_{p_i}$ or $\mathscr{F} \subseteq \mathscr{P}_{p_i}$ for all $i \in [m]$.*

*Otherwise the problem is $\#_p P$-hard for some $p|k$. More specific, we have*

- *For $i \in [m]$, if $\mathscr{F} \not\subseteq \mathscr{A}_{p_i}$ and $\mathscr{F} \not\subseteq \mathscr{P}_{p_i}$, then $\#_k CSP(\mathscr{F})$ is $\#_{p_i} P$-hard.*
- *If $r_0 = 1$, $\mathscr{F} \not\subseteq \mathscr{A}_2$, and it is not the case that every function in $\mathscr{F}$ are closed under complement after mod 2, then $\#_k CSP(\mathscr{F})$ is $\#_2 P$-hard.*
- *If $r_0 \geq 2$ and $\mathscr{F} \not\subseteq \mathscr{A}_2$, then $\#_k CSP(\mathscr{F})$ is $\#_2 P$-hard.*

The statement of the main theory is a little complicated due to technique reason [2] . In terms of dichotomy, we have a simple statement.

▶ **Theorem 20.** *Let $k > 1$ and $\mathscr{F}$ be a set of functions. Then $\#_k CSP(\mathscr{F})$ is either in P or $\#_p P$-hard for some $p|k$.*

**Acknowledgements** Work partly done when Heng Guo was a master student in Peking University, and Sangxia Huang was an intern student at Microsoft Research Asia and an undergraduate student of Shanghai Jiao Tong University. Sangxia Huang is supported by ERC Advanced investigator grant 226203. Mingji Xia is supported by NSFC 61003030 and 60970003, the CAS start-up fund for CAS President Scholarship winner, the Hundred-Talent program of Chinese Academy of Sciences under Angsheng Li, and the Grand Challenge Program "Network Algorithms and Digital Information" of ISCAS.

## References

1 Richard Beigel and John Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theor. Comput. Sci.*, 103(1):3–23, 1992.

2 A. Bulatov, M. Dyer, L.A. Goldberg, M. Jalsenius, and D. Richerby. The complexity of weighted boolean #CSP with mixed signs. *Theoretical Computer Science*, 410(38-40):3949–3961, 2009.

3 Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006.

4 Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (1)*, volume 5125 of *Lecture Notes in Computer Science*, pages 646–661. Springer, 2008.

5 Andrei A. Bulatov and Víctor Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. In *FOCS*, pages 562–571. IEEE Computer Society, 2003.

6 Andrei A. Bulatov and Martin Grohe. The complexity of partition functions. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 294–306. Springer, 2004.

[2] The statement of our dichotomy is slightly different with Faben's [14]. The dichotomy by Faben stated that a problem is either P or $\#_k$P-hard. This is because in the unweighted case, the criterion for different odd prime $p$ is identical.

7 Jin-Yi Cai, Xi Chen, and Pinyan Lu. Graph homomorphisms with complex values: A dichotomy theorem. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP (1)*, volume 6198 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 2010.

8 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holant problems and counting CSP. In Michael Mitzenmacher, editor, *STOC*, pages 715–724. ACM, 2009.

9 N. Creignou, S. Khanna, and M. Sudan. *Complexity classifications of boolean constraint satisfaction problems*. SIAM Monographs on Discrete Mathematics and Applications, 2001.

10 Nadia Creignou and Miki Hermann. Complexity of generalized satisfiability counting problems. *Inf. Comput.*, 125(1):1–12, 1996.

11 Martin E. Dyer, Leslie Ann Goldberg, and Mark Jerrum. The complexity of weighted boolean #CSP. *SIAM J. Comput.*, 38(5):1970–1986, 2009.

12 Martin E. Dyer, Leslie Ann Goldberg, and Mike Paterson. On counting homomorphisms to directed acyclic graphs. *J. ACM*, 54(6), 2007.

13 Martin E. Dyer and Catherine S. Greenhill. The complexity of counting graph homomorphisms. *Random Struct. Algorithms*, 17(3-4):260–289, 2000.

14 John Faben. The complexity of counting solutions to generalised satisfiability problems modulo k. *CoRR*, abs/0809.1836, 2008.

15 Leslie Ann Goldberg, Martin Grohe, Mark Jerrum, and Marc Thurley. A complexity dichotomy for partition functions with mixed signs. In *STACS*, pages 493–504, 2009.

16 R. Hartshorne. *Algebraic geometry*. springer Verlag, 1977.

17 Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

18 Christos H. Papadimitriou and Stathis Zachos. Two remarks on the power of counting. In *Proceedings of the 6th GI-Conference on Theoretical Computer Science*, pages 269–276, 1982.

19 T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, page 226. ACM, 1978.

20 Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

21 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

22 Leslie G. Valiant. Accidental algorthims. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 509–517, Washington, DC, USA, 2006. IEEE Computer Society.

23 Leslie G. Valiant. Some observations on holographic algorithms. In Alejandro López-Ortiz, editor, *LATIN*, volume 6034 of *Lecture Notes in Computer Science*, pages 577–590. Springer, 2010.


## A Proof of Lemma 14

**Proof.** Suppose $F$ has arity $r$. Since $\mathscr{P}$ contains all unary functions and $F \notin \mathscr{P}$, $r \geq 2$. Define an $|R_F| \times r$ $\{0,1\}$-matrix whose rows list every element of $R_F$, and columns correspond to $x_1, \ldots, x_r$.

We first remove any column which is all-0 or all-1 and update the table to $R_{F^{x_i=0}}$ or $R_{F^{x_i=1}}$, respectively. If two columns are identical or are complementary in every bit, we remove one of them and update the table to $R_{F^{x_j=*}}$, where $j$ corresponds to the column removed. We remove columns as long as possible. It is easy to see that this removal process maintains the property of not belonging to $\mathscr{P}$.

Now we suppose there is some $G \notin \mathscr{P}$ where no more columns can be removed by the above process. There must be some columns left in the table, otherwise the function just before the last column removal is a unary function, hence in $\mathscr{P}$. In fact $G$ being not in $\mathscr{P}$, the arity of $G$ is $\geq 2$. For simplicity we still denote it by $r$. We have two cases:

**Case 1:** $|R_G| < 2^r$**.** By Lemma 12, we may assume $R_G$ is affine, given by an affine linear system $AX = 0$. We have shown that $|R_G| \neq 0$, as some columns remain. Since $G$ is not unary, the table has more than one columns. If $|R_G| = 1$, any two columns (of length one) must be identical or complementary and the removal process should have continued. Thus $|R_G| > 1$. W.l.o.g. assume $x_1, \ldots, x_s$ are free variables in $AX = 0$ and $x_{s+1}, \ldots, x_k$ are dependent variables. $|R_G| = 2^s$ is a power of 2. We have shown that $s \geq 1$. By $|R_G| < 2^r$, $s < r$. We claim $s \geq 2$. If instead $s = 1$, then every $x_2, \ldots, x_r$ is dependent on $x_1$ on $R_G$, so the column at $x_2$ must be an all-0 or all-1 column, or be identical or complementary to $x_1$. The expression of $x_r$ in terms of $x_1, \ldots, x_s$ must involve at least two non-zero coefficients; otherwise the column at $x_r$ must be an all-0 or all-1 column, or be identical or complementary to another column. W.l.o.g., say the coefficients of $x_1, x_2$ are non-zero.

Let $P(x_1, x_2, x_r) = G^{x_3=0,\ldots,x_s=0,x_{s+1}=*,\ldots,x_{r-1}=*}$ (these two lists of variables could be empty). It can be verified that $R_P = \chi_{x_1 \oplus x_2 \oplus x_r = c}$ for some $c \in \mathbb{Z}_2$.

The affine linear equation $x_1 \oplus x_2 \oplus x_r = c$ is symmetric. Now we define a "symmetrized" function $H(x_1, x_2, x_r) = \prod_{\sigma \in S_3} P(x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(r)})$, where $S_3$ is the symmetry group on three letters $\{1, 2, k\}$. This $H$ is a symmetric function on $(x_1, x_2, x_r)$ and has support $R_H = R_P$. Thus, after normalizing, $H = [a, 0, 1, 0]$ or $[0, 1, 0, a]$ where $a \neq 0$. We remark that this ternary function $H \notin \mathscr{P}$.

**Case 2:** $|R_G| = 2^r$**.** If for all $1 \leq i \leq r$, the ratio $G^{x_i=1}/G^{x_i=0}$ is a constant function $c_i$, (since $|R_G| = 2^r$ there are no divisions by zeros), then $G = c_0 \cdot \prod_{1 \leq i \leq r} U_i(c_i)$, where the constant $c_0 = G^{x_1=0,\ldots,x_r=0}$, and $U_i(c_i)$ is the unary function $[1, c_i]$ on $x_i$. This gives $G \in \mathscr{P}$, a contradiction.

Now suppose for some $i$, $G^{x_i=1}/G^{x_i=0}$ is not a constant function. W.l.o.g., assume that $i = 1$. The Boolean hypercube on $(x_2, \ldots, x_r) \in \{0, 1\}^{r-1}$ is connected by edges which flip just one bit. W.l.o.g., suppose that
$G^{x_1=1}/G^{x_1=0}(0, a_3, \ldots, a_r) \neq G^{x_1=1}/G^{x_1=0}(1, a_3, \ldots, a_r)$. Set $x_3 = a_3, \ldots, x_r = a_r$, we get a binary function $H(x_1, x_2) = G(x_1, x_2, a_3, \ldots, a_r)$. We have that $H(1, 0)/H(0, 0) \neq H(1, 1)/H(0, 1)$, hence the rank of $H = \begin{bmatrix} H(0,0) & H(0,1) \\ H(1,0) & H(1,1) \end{bmatrix}$ is 2.

If $H$ were in $\mathscr{P}$, then partition the variable set according to connectivity by binary equality and disequality functions. If any connected component has at least 2 variables, we can set values to these 2 variables so that $H = 0$. But $H$ is never zero. Then each component must be a single variable and $H$ is defined by a product of unary functions. But such a function has rank 1. This contradiction completes our proof. ◀